**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO BÀI TẬP LỚN**

**HỌC PHẦN: THỰC TẬP CƠ SỞ**

**MÃ HỌC PHẦN: INT13147**

**ĐỀ TÀI: Xây dựng website tìm kiếm việc làm**

**Giảng viên hướng dẫn : Ths.Bùi Văn Kiên**

**Nhóm : 10**

**Sinh viên thực hiện : Đỗ Gia Phong - B22DCCN613**

**HÀ NỘI 2025**

**MỤC LỤC**

# GIới thiệu tổng quan về dự án

## Giới thiệu dự án

Trong bối cảnh nền kinh tế đang không ngừng phát triển và chuyển mình theo xu hướng công nghệ hóa, số hóa, nhu cầu tìm kiếm việc làm của người lao động cũng như nhu cầu tuyển dụng của các doanh nghiệp ngày càng tăng cao và trở nên đa dạng. Tuy nhiên, việc kết nối giữa người tìm việc và nhà tuyển dụng vẫn còn tồn tại nhiều khó khăn, đặc biệt là ở những khu vực chưa được tiếp cận đầy đủ với các kênh thông tin hiện đại. Các phương pháp truyền thống như phát tờ rơi, thông báo trực tiếp hay đăng tin trên báo giấy đang dần trở nên lỗi thời, không còn đáp ứng được yêu cầu về tốc độ, tính chính xác và khả năng cập nhật liên tục của thị trường lao động hiện đại.

Trong những năm gần đây, sự phổ biến của internet và các nền tảng web đã tạo điều kiện thuận lợi cho việc phát triển các hệ thống hỗ trợ tuyển dụng và tìm kiếm việc làm trực tuyến. Thông qua các trang web, người dùng có thể dễ dàng tạo hồ sơ cá nhân, tìm kiếm công việc phù hợp theo ngành nghề, vị trí địa lý, mức lương, kỹ năng yêu cầu và nhiều tiêu chí khác. Đồng thời, các nhà tuyển dụng cũng có thể nhanh chóng tiếp cận nguồn ứng viên tiềm năng, đăng tin tuyển dụng và quản lý hồ sơ ứng tuyển một cách hiệu quả.

Dựa trên thực tế đó, việc xây dựng một **website tìm kiếm việc làm** đóng vai trò vô cùng quan trọng và cấp thiết, nhằm tạo ra một cầu nối hiệu quả giữa người lao động và doanh nghiệp. Hệ thống không chỉ giúp giảm thiểu chi phí và thời gian cho cả hai bên mà còn tạo điều kiện cho người tìm việc tiếp cận được nhiều cơ hội việc làm phù hợp, đồng thời hỗ trợ doanh nghiệp nâng cao hiệu quả trong quá trình tuyển dụng.

Đề tài “Xây dựng website tìm kiếm việc làm” được thực hiện nhằm mục tiêu tạo ra một nền tảng trực tuyến thân thiện, dễ sử dụng, tích hợp đầy đủ các tính năng cần thiết như tìm kiếm công việc theo tiêu chí, đăng ký tài khoản, tạo hồ sơ cá nhân, ứng tuyển trực tuyến, quản lý tin tuyển dụng,... Bên cạnh đó, hệ thống cũng đảm bảo tính bảo mật thông tin người dùng và khả năng mở rộng trong tương lai để phù hợp với sự phát triển của thị trường lao động.

## Tổng quan hệ thống

### Kiến trúc tổng quan

Hệ thống được xây dựng theo mô hình **Monolithic** , với các thành phần chính:

**FrontEnd :**

* Sử dụng HTML, CSS, JS, Bootstrap để xây dựng giao diện.
* Kết nối với API từ backend để hiển thị dữ liệu theo thời gian thực.

**Backend :**

* Xây dựng bằng Spring Boot (Java)
* Cung cấp API RESTful cho frontend.

**Database :**

* PostgreSQL để lưu trữ thông tin người dùng, ứng viên, công việc.
* Redis lưu trữ người dùng nhận thông báo

## Chi tiết về công nghệ sử dụng

### Front End

**HTML (HyperText Markup Language)**

* HTML ngôn ngữ đánh dấu dùng để xây dựng cấu trúc cơ bản của một trang web.
* Chức năng chính :
  + Định nghĩa các phần tử như: tiêu đề, đoạn văn, hình ảnh, biểu mẫu, nút bấm, bảng, liên kết, v.v.
  + Là "bộ xương" của giao diện người dùng, giúp trình duyệt hiểu được nội dung cần hiển thị.

**CSS (Cascading Style Sheets)**

* CSS là ngôn ngữ dùng để tạo kiểu dáng, định dạng và làm đẹp cho trang web.
* Chức năng chính :
  + Quy định màu sắc, phông chữ, bố cục, độ rộng, khoảng cách, hiệu ứng chuyển động,...
  + Giúp giao diện trở nên trực quan và dễ sử dụng hơn.

**JS (Javascript)**

* Ngôn ngữ lập trình phía client giúp xử lý logic, tương tác, và hành vi động của website.
* Chức năng chính :
  + Bắt sự kiện người dùng (click, nhập dữ liệu, cuộn trang,...)
  + Giao tiếp với backend (qua API)
  + Thay đổi nội dung trang web mà không cần tải lại (AJAX)

**Bootstrap 5**

* Framework CSS & JS phổ biến giúp xây dựng giao diện web nhanh chóng, đẹp và tương thích đa thiết bị (responsive).
* Chức năng chính :
  + Bộ **component sẵn có**: navbar, button, form, card, modal,...
  + **Lưới 12 cột (Grid system):** dễ dàng bố trí giao diện responsive.
  + **Tùy chỉnh nhanh:** thông qua class mà không cần viết CSS nhiều.
* Ưu điểm :
  + Giảm thời gian phát triển.
  + Hỗ trợ tốt trên mọi trình duyệt.
  + Tương thích tốt với thiết bị di động.

### BackEnd

**Ngôn ngữ lập trình Java**

Java là một ngôn ngữ lập trình hướng đối tượng được giới thiệu bởi Sun Microsystems vào năm 1995. Đặc điểm nổi bật của Java là nguyên tắc "Viết một lần, chạy mọi nơi" (WORA), cho phép lập trình viên viết mã mà không cần phải lo lắng về nền tảng cụ thể, vì mã sẽ được biên dịch thành mã bytecode có thể chạy trên mọi thiết bị có Java Virtual Machine (JVM). Ngôn ngữ này cũng tự động quản lý bộ nhớ thông qua cơ chế thu gom rác, giúp giảm thiểu rủi ro khi lập trình viên phải quản lý bộ nhớ. Hơn nữa, Java cung cấp một bộ thư viện phong phú hỗ trợ cho việc phát triển nhiều loại ứng dụng, từ ứng dụng web và di động (như ứng dụng Android) đến các hệ thống lớn phục vụ cho doanh nghiệp. Nhờ vào cộng đồng lập trình viên đông đảo và các tài nguyên phong phú, Java vẫn giữ được vị trí quan trọng trong lĩnh vực công nghệ thông tin, là sự lựa chọn phổ biến cho nhiều dự án phần mềm hiện nay.

**Framework Spring Boot**

Spring Boot là một **framework mã nguồn mở** được xây dựng dựa trên **Spring Framework**, giúp phát triển ứng dụng Java một cách **nhanh chóng, đơn giản và dễ cấu hình** hơn. Nó được phát triển bởi Pivotal Team. Spring Boot giúp việc tạo ra các ứng dụng Spring độc lập và sẵn sàng cho môi trường sản xuất trở nên dễ dàng.

Spring Boot cung cấp một loạt các tính năng và lợi ích sau :

* Linh hoạt: Spring Boot cung cấp nhiều cách linh hoạt để cấu hình Java Beans, cấu hình XML và giao dịch cơ sở dữ liệu.
* Xử lý Batch mạnh mẽ: Spring Boot cung cấp cơ chế batch mạnh mẽ.
* Hỗ trợ Microservices: Micro Service là một kiến trúc cho phép các nhà phát triển phát triển và triển khai dịch vụ một cách độc lập. Mỗi dịch vụ chạy trong một quá trình riêng biệt, giúp đạt được mô hình nhẹ nhàng hỗ trợ các ứng dụng doanh nghiệp. Spring Boot cung cấp cơ chế để phát triển và kiểm tra các microservices một cách dễ dàng.
* Cấu hình tự động: Trong Spring Boot, mọi thứ đều được cấu hình tự động; không cần cấu hình thủ công.
* Dựa trên Annotation: Trong Spring Boot, chúng ta có thể tạo một ứng dụng chạy với rất ít annotation.
* Quản lý phụ thuộc dễ dàng: Spring Boot cung cấp nhiều starter theo nhu cầu như cho web, cho cơ sở dữ liệu, v.v. để quản lý phụ thuộc một cách hiệu quả. Một starter project cung cấp quản lý phụ thuộc cho các chức năng tương ứng.
* Container Servlet nhúng: Spring Boot cung cấp một container servlet nhúng (Jetty) có thể thay đổi. Container này rất hữu ích khi thử nghiệm ứng dụng. Chúng ta có thể thử nghiệm tất cả các chức năng mà không cần triển khai ứng dụng Spring Boot trên bất kỳ container ứng dụng bên ngoài nào.

**Thư viện sử dụng**

**Spring Data JPA**:

* Spring Data JPA là một phần mở rộng của Spring Data, hỗ trợ tương tác với cơ sở dữ liệu bằng cách cung cấp các API dựa trên JPA (Java Persistence API).
* Nó giúp giảm bớt việc viết mã SQL thủ công, thay vào đó cung cấp các phương thức truy vấn đơn giản với cú pháp rõ ràng và dễ sử dụng.
* Ta có thể tạo các repository để thao tác CRUD (tạo, đọc, cập nhật, xóa) mà không cần viết nhiều mã SQL phức tạp.

**Lombok**:

* Lombok là một thư viện giúp giảm bớt mã boilerplate (mã không cần thiết) trong Java.
* Bằng cách sử dụng các annotation của Lombok như @Getter, @Setter, @Data, và @Builder, ta có thể tự động sinh mã cho các phương thức getter, setter, constructor, builder và nhiều tính năng khác.
* Điều này giúp mã nguồn trở nên ngắn gọn và dễ đọc hơn.

**MapStruct:**

* MapStruct là một thư viện Java dùng để ánh xạ (mapping) giữa các đối tượng DTO và entity một cách tự động.
* Nó giúp chuyển đổi dữ liệu giữa các lớp (classes) một cách nhanh chóng và tránh các lỗi thường gặp trong việc ánh xạ thủ công.
* Bằng cách sử dụng các interface và annotation của MapStruct, bạn có thể định nghĩa các ánh xạ mà không cần viết nhiều mã chuyển đổi.

**Hibernate:**

* Hibernate là một framework ORM (Object-Relational Mapping) phổ biến nhất trong Java, được sử dụng để ánh xạ các lớp Java với các bảng trong cơ sở dữ liệu.
* Nó giúp quản lý các giao dịch, hỗ trợ lazy loading, caching và cung cấp các tính năng như xử lý quan hệ (One-to-Many, Many-to-Many) một cách linh hoạt.
* Hibernate thường được sử dụng cùng với Spring Data JPA, giúp quản lý và truy vấn dữ liệu một cách thuận tiện hơn.

**PostgresSql**

* **PostgresSQL** là một trình điều khiển (driver) JDBC giúp Spring Boot kết nối và tương tác với cơ sở dữ liệu PostgresSQL.
* Khi tích hợp PostgresSQL, ta có thể sử dụng các thư viện như Spring Data JPA và Hibernate để thực hiện các thao tác CRUD, truy vấn dữ liệu, và quản lý giao dịch trên Postgres.
* Driver này giúp chuyển đổi các câu lệnh SQL từ ứng dụng Java thành các lệnh mà Postgres hiểu và thực thi, làm cho việc giao tiếp với cơ sở dữ liệu trở nên liền mạch.

**Cloudinary:**

* Cloudinary là một dịch vụ lưu trữ và quản lý phương tiện truyền thông (media) đám mây, thường được sử dụng để lưu trữ, xử lý và phân phối hình ảnh, video.
* Khi tích hợp Cloudinary, ta có thể tải ảnh và video lên từ ứng dụng của mình và nhận lại URL để truy cập hoặc hiển thị.
* Cloudinary còn cung cấp các API để xử lý ảnh và video, ví dụ như thay đổi kích thước, cắt, nén ảnh, và chuyển đổi định dạng, giúp việc quản lý hình ảnh trong ứng dụng của mình trở nên dễ dàng và hiệu quả.

**WebPush :**

* web-push là thư viện dùng để gửi thông báo đẩy (push notification) từ server đến trình duyệt.
* Hỗ trợ tạo và sử dụng VAPID keys để xác thực.
* Dùng để gửi thông báo real-time tới người dùng đã đăng ký, ngay cả khi họ không mở trang web.
* Phù hợp cho các tính năng như: thông báo đơn hàng, tin nhắn, cảnh báo hệ thống,...

**WebSocket**

* WebSocket là một giao thức mạng giúp thiết lập kết nối hai chiều liên tục (full-duplex) giữa client và server, thường được sử dụng trong các ứng dụng cần cập nhật dữ liệu theo thời gian thực như chat, thông báo, hoặc hệ thống giám sát.
* Khác với HTTP (mỗi lần gửi là một lần kết nối), WebSocket cho phép:
  + Server và client giữ kết nối lâu dài và gửi/nhận dữ liệu bất kỳ lúc nào.
  + Giảm độ trễ, tiết kiệm tài nguyên do không cần thiết lập lại kết nối liên tục.
  + Tăng hiệu quả truyền tải dữ liệu trong các ứng dụng cần phản hồi nhanh.

## Kết chương

# Thiết kế hệ thống

## Tổng quan nghiệp vụ

### Phân hệ User (Người dùng)

Ứng viên là người dùng chính của hệ thống, có thể :

**Xác thực tài khoản :**

* Đăng ký tài khoản bằng username/password hoặc Google OAuth.
* Đăng nhập, đăng xuất, lấy lại mật khẩu khi quên

**Tìm kiếm và quản lý công việc :**

* Tìm kiếm việc làm theo từ khóa, ngành nghề, địa điểm.
* **Xem chi tiết công việc** :Bao gồm mô tả, yêu cầu, lương, địa điểm, công ty đăng tuyển.
* **Ứng tuyển công việc** : Theo dõi trạng thái ứng tuyển (**thành công, bị từ chối**).
* Quản lý danh sách **công việc yêu thích**.

**Cá nhân hóa thông tin**

* Chỉnh sửa hồ sơ cá nhân, cập nhật kỹ năng, kinh nghiệm.

**Nhận thông báo :**

* Xem danh sách thông báo hệ thống.
* Nhận **thông báo đẩy** về việc tuyển dụng, trạng thái ứng tuyển.

### Phân hệ Recruiter (Nhà tuyển dụng)

Nhà tuyển dụng có thể:

**Quản lý tin tuyển dụng**:

* Thêm, chỉnh sửa, xóa **công việc tuyển dụng**.
* Xem danh sách công việc đang tuyển.

**Quản lý ứng viên**:

* Xem danh sách ứng viên đã ứng tuyển.
* **Phê duyệt hoặc từ chối** ứng viên.

**Cập nhật thông tin công ty**:

* Chỉnh sửa thông tin công ty hiển thị trên hệ thống.

**Nhận thông báo :**

* Xem danh sách thông báo hệ thống.
* Nhận **thông báo đẩy** về việc tuyển dụng, trạng thái ứng tuyển.

### Phân hệ Admin (Quản trị viên)

Quản trị viên chịu trách nhiệm giám sát và vận hành hệ thống:

**Quản lý người dùng:**

* Xem danh sách ứng viên, nhà tuyển dụng.
* Xóa tài khoản vi phạm.

**Quản lý công việc**:

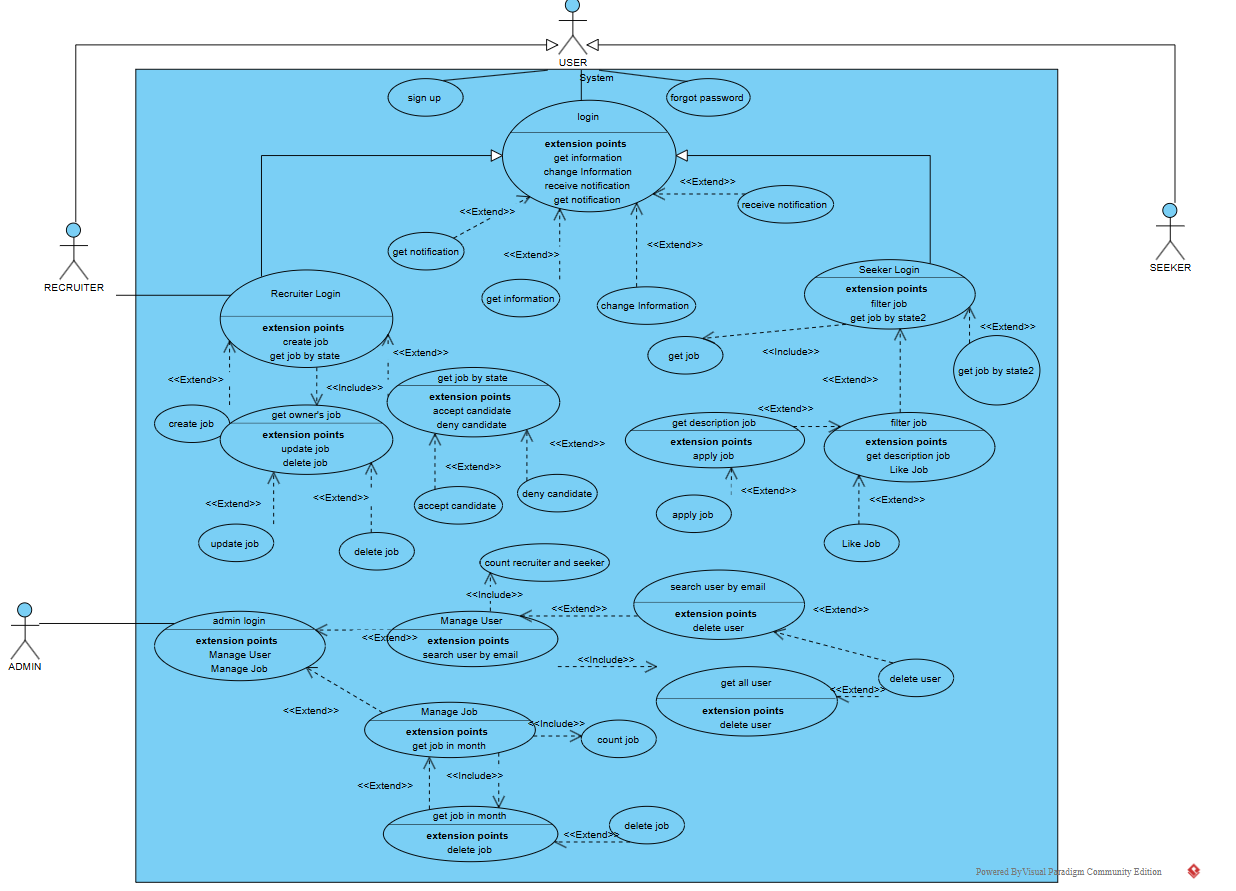
* Giám sát danh sách công việc đang được tuyển dụng.

**Phân tích & báo cáo**:

* Xem **thống kê theo ngành nghề, số lượng ứng viên, tỉ lệ ứng tuyển thành công**.

## Phân tích yêu cầu chức năng

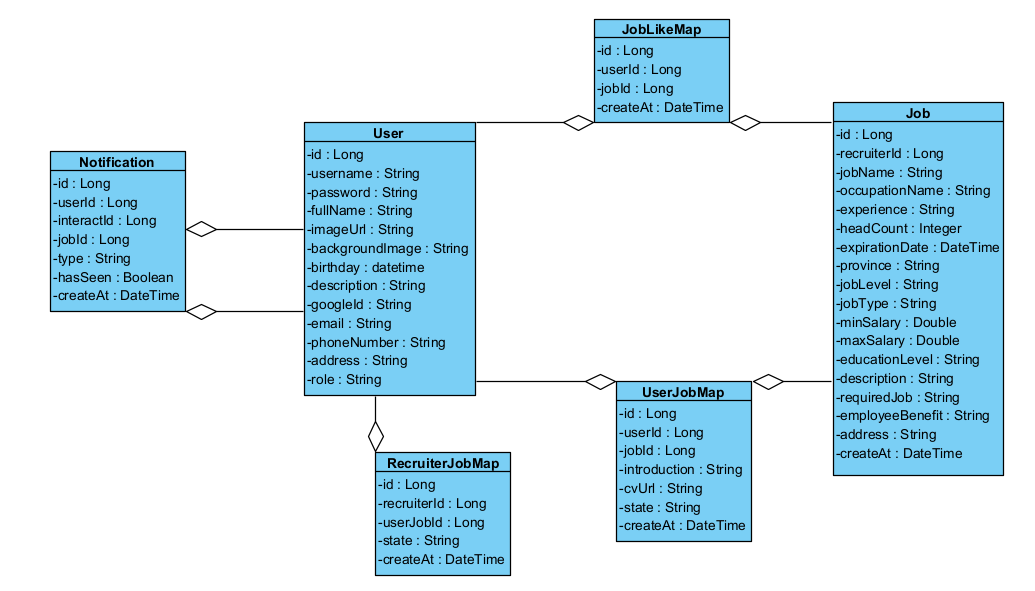
### Tổng quan hệ thống



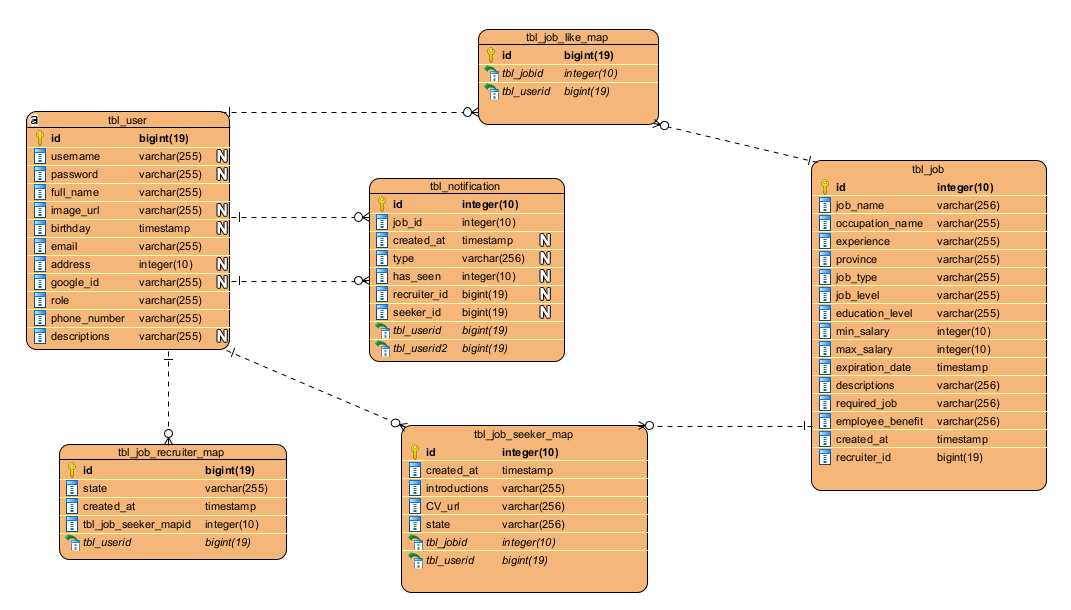
Các usecase được mô tả chi tiết :

* Sign up : Usecase này cho phép seeker, recruiter đăng ký tài khoản
* Login : Usecase này cho phép seeler, recruiter, admin đăng nhập vào hệ thống website
* Forgot password : Usecase này cho phép seeker, recruiter lấy lại mật khẩu đã quên
* Get information : cho phép seeker, recruiter xem thông tin cá nhân
* Change Information : cho phép seeker, recruiter thay đổi thông tin
* Recieve notification : cho phép seeker, recruiter nhận thông báo đẩy
* Get notification : cho phép seeker, recruiter xem thông báo
* Get job by state : cho phép seeker, recruiter xem công việc theo từng trạng thái đơn ứng tuyển
* Get job : cho phép người tìm việc xem các công việc được đăng tuyển bởi các nhà tuyển dụng
* Filter job : cho phép người tìm việc tìm kiếm công việc theo tiêu chí nào đó
* Get descriptions job : cho phép người tìm việc xem chi tiết công việc đang tuyển dụng
* Apply job : cho phép người tìm việc nộp đơn ứng tuyển cho công việc
* Get Owner’s job : cho phép nhà tuyển dụng xem được các công việc mình đang đăng tuyển
* Create Job : cho phép nhà tuyển dụng đăng tải thông tin tuyển dụng
* Update Job : cho phép nhà tuyển dụng thay đổi thông tin tuyển dụng
* Delete Job : cho phép nhà tuyển dụng xóa công việc tuyển dụng
* Accept candidate : cho phép nhà tuyển dụng phê duyệt ứng viên
* Deny candidate : cho phép nhà tuyển dụng từ chối ứng viên
* Manage User : cho phép admin quản lý người dùng
* Manage Job : cho phép admin quản lý công việc.

## Biểu đồ lớp thực thể



## Thiết kế cơ sở dữ liệu



Hệ thống tuyển dụng sử dụng cơ sở dữ liệu gồm nhiều bảng để quản lý thông tin người dùng, công việc và các hoạt động tương tác giữa người tìm việc và nhà tuyển dụng. Trung tâm của hệ thống là bảng **tbl\_user**, nơi lưu trữ thông tin đăng nhập và hồ sơ cơ bản của tất cả người dùng. Mỗi người dùng có thể đóng vai trò là **người tìm việc (seeker)** hoặc **nhà tuyển dụng (recruiter).**

Nhà tuyển dụng có thể đăng nhiều công việc, và các công việc này được lưu trong bảng **tbl\_job.** Mỗi bản ghi trong **tbl\_job** chứa thông tin mô tả công việc, yêu cầu ứng viên và mức lương đề xuất. Mối quan hệ giữa **tbl\_recruiter** và **tbl\_job** là **một-nhiều** (một recruiter đăng nhiều job).

Người tìm việc có thể nộp đơn vào nhiều công việc thông qua bảng **tbl\_job\_seeker\_map**. Bảng này đóng vai trò trung gian ghi nhận mối quan hệ giữa người dùng và các công việc mà họ ứng tuyển, kèm theo thời gian nộp, lời giới thiệu và đường dẫn CV. Mỗi bản ghi trong bảng này có thể nhận được phản hồi từ nhà tuyển dụng, và phản hồi đó được lưu trong bảng **tbl\_job\_recruiter\_map**. Bảng này ghi lại trạng thái phản hồi của recruiter như “đã đọc”, “chấp nhận”, “từ chối”... Mối quan hệ từ **tbl\_job\_seeker\_map** đến **tbl\_job\_recruiter\_map** là **một-một**.

Ngoài ứng tuyển, người dùng còn có thể **yêu thích** (like) một công việc thông qua bảng **tbl\_job\_like\_map**. Đây là bảng trung gian thiết lập quan hệ **nhiều-nhiều** giữa **tbl\_user** và **tbl\_job**.

Hệ thống còn hỗ trợ chức năng **theo dõi** giữa recruiter và seeker thông qua bảng **tbl\_follower**. Bảng này ghi nhận những trường hợp recruiter theo dõi seeker, tạo nên mối quan hệ **nhiều-nhiều** giữa hai đối tượng này. Điều này giúp recruiter dễ dàng theo dõi các seeker tiềm năng.

Cuối cùng, để quản lý các tương tác trong hệ thống, bảng **tbl\_notification** được sử dụng để lưu trữ các thông báo giữa recruiter và seeker, như thông báo về trạng thái hồ sơ, lời mời phỏng vấn hoặc phản hồi từ nhà tuyển dụng. Mỗi thông báo có thể liên kết với một công việc cụ thể, người gửi và người nhận, và trạng thái đã xem hay chưa.

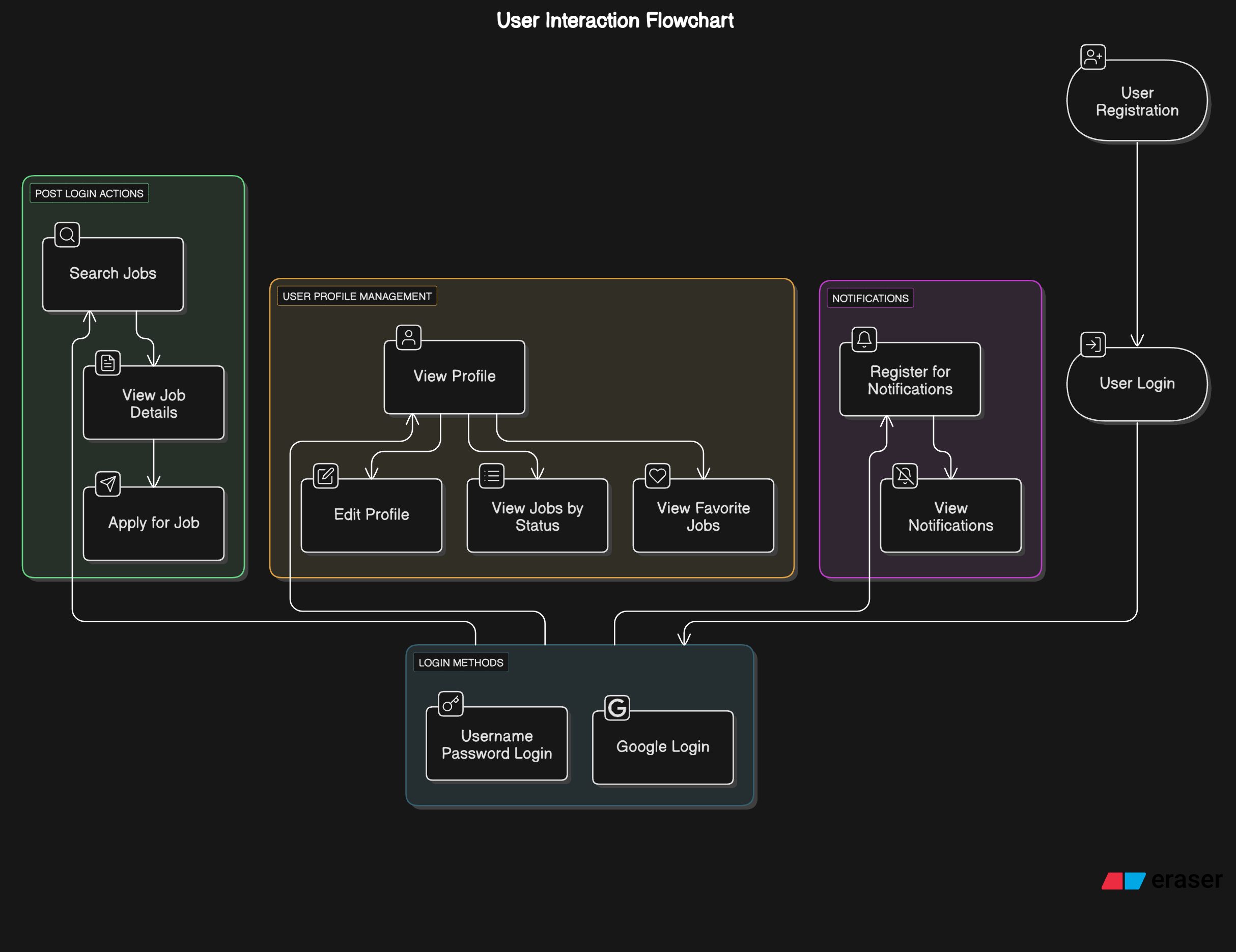
Tổng thể, cơ sở dữ liệu được thiết kế xoay quanh người dùng, cho phép tương tác đa chiều giữa seeker và recruiter thông qua ứng tuyển, phản hồi, yêu thích, theo dõi và thông báo. Cách thiết kế này giúp hệ thống linh hoạt và dễ mở rộng, đồng thời hỗ trợ đầy đủ các chức năng cần thiết cho một nền tảng tuyển dụng chuyên nghiệp.

# Thiết kế chi tiết

## Quy trình hoạt động của hệ thống

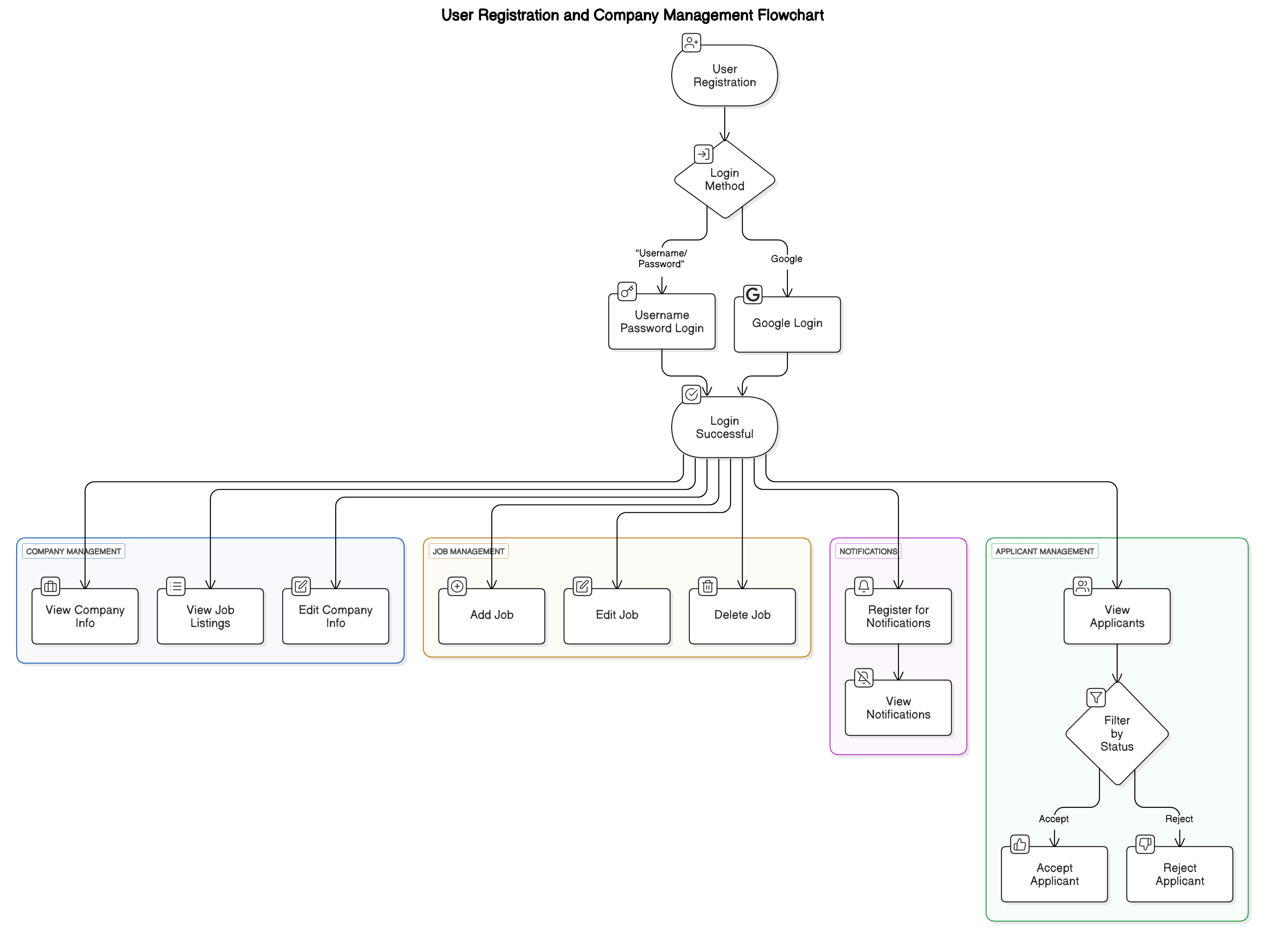
### Quy trình ứng viên tìm việc

* Đăng nhập / Đăng ký tài khoản.
* Tìm kiếm công việc theo ngành nghề, địa điểm, mức lương.
* Xem chi tiết công việc và nộp đơn ứng tuyển.
* Theo dõi trạng thái ứng tuyển (đang xét duyệt, thành công, bị từ chối).
* Nhận thông báo khi có cập nhật từ nhà tuyển dụng.



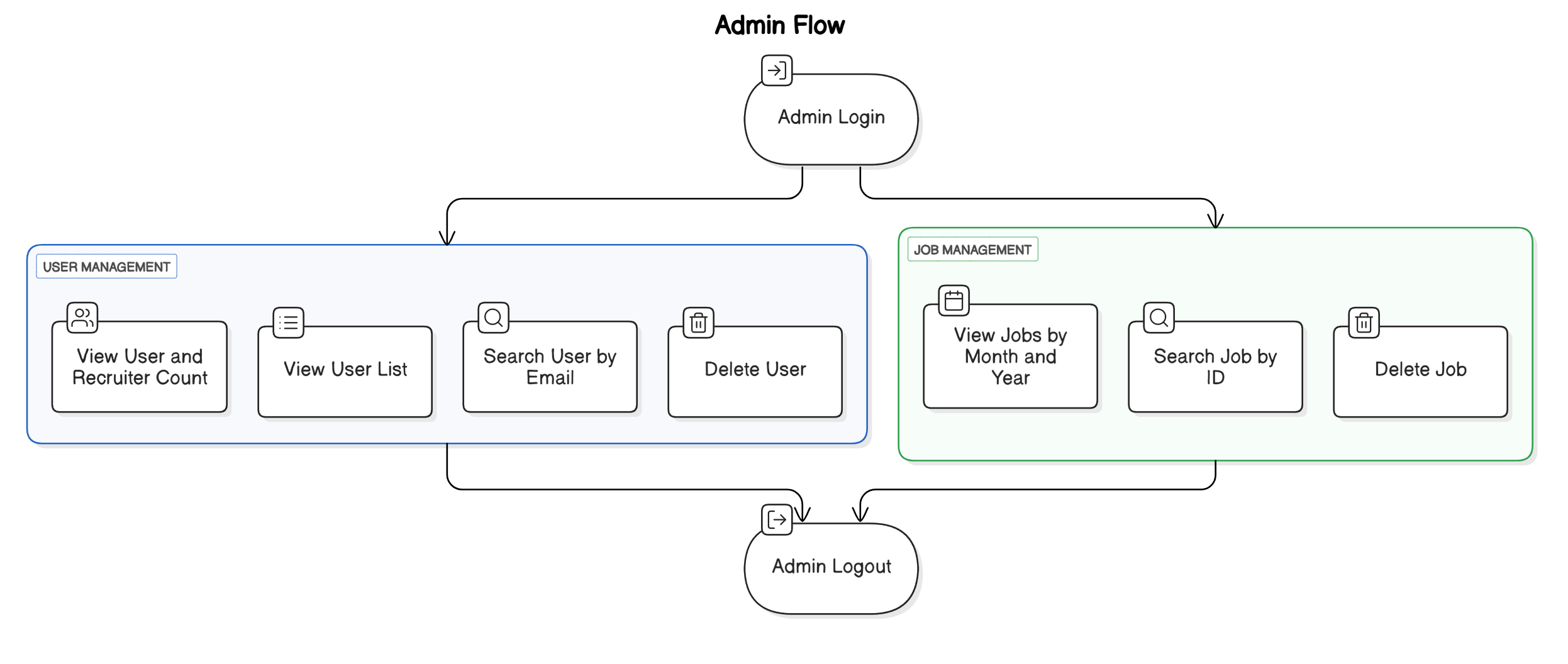
### Quy trình nhà tuyển dụng đăng tuyển

* Đăng nhập tài khoản nhà tuyển dụng.
* Thêm công việc mới, điền thông tin mô tả, yêu cầu, mức lương.
* Quản lý danh sách công việc đang tuyển.
* Xem danh sách ứng viên, phê duyệt hoặc từ chối.
* Cập nhật thông tin công ty để tăng độ tin cậy.



### Quy trình quản trị viên

* Quản lý người dùng, kiểm tra tài khoản vi phạm.
* Giám sát công việc.
* Xem báo cáo phân tích, hỗ trợ cải thiện trải nghiệm người dùng.



## Mô tả tính năng cụ thể

### Thông tin chung api

#### Thông tin chung

|  |  |
| --- | --- |
| Base URL | http://localhost:8086/api |
| Version | v1 |
| Authentication | Bearer Token |

#### Định dạng phản hồi JSON

1. Yêu cầu thành công(2xx):

|  |  |  |
| --- | --- | --- |
| Tên trường | Kiểu | Mô tả |
| code | String | Mã trạng thái |
| message | String | Mô tả đi kèm |
| data | Object | Dữ liệu trả về |

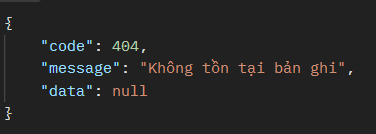
**VD :**



1. Yêu cầu thất bại :

|  |  |  |
| --- | --- | --- |
| Tên trường | Kiểu | Mô tả |
| code | String | Mã trạng thái |
| message | String | Mô tả đi kèm |
| data | Null | Không có dữ liệu |

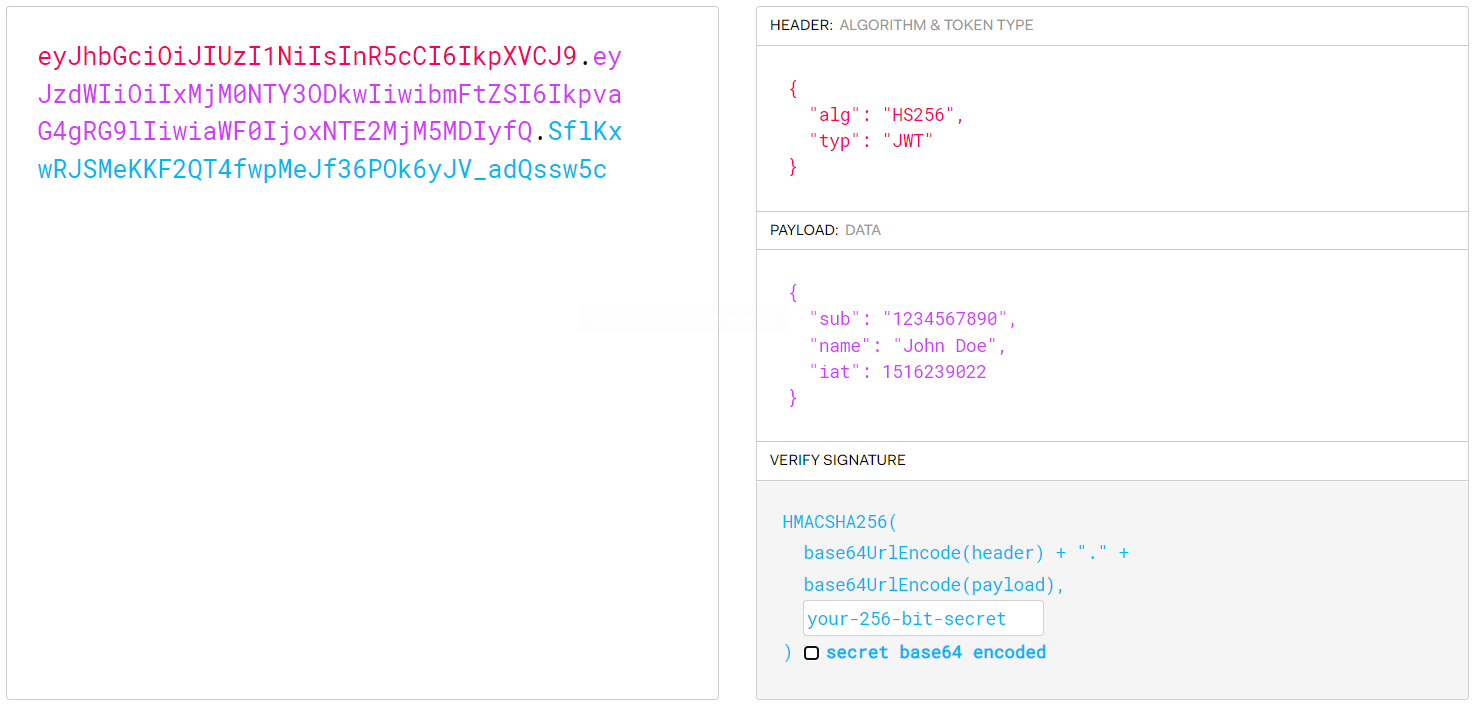
**VD :**



### Đăng ký, đăng nhập

1. JSON Web Token

* JSON Web Token là chuỗi mã hóa được sinh ra sau khi người dùng đăng nhập vào ứng dụng, chuỗi mã hóa này được sử dụng để xác minh quyền truy cập vào tài nguyên mà không cần nhập lại username/password nữa.
* Cấu trúc của JWT :
  + Header : gồm 2 phần chính :
    - typ - Loại token (mặc định là JWT - cho biết đây là 1 token JWT)
    - Alg – Là thuật toán dùng để mã hóa.
* Từ 2 thành phần trên thì header sẽ được mã hóa dựa trên mã hóa base64url để nhận được đoạn mã bất kì
  + Payload : Là nơi chứa các nội dụng thông tin, nó có thể là mô tả 1 thực thể hoặc cũng là cá thông tin bổ sung cho header, VD có thể 1 vài thông tin người dùng hoặc ngày bắt đầu, ngày hết hạn ... Sau đó nó cũng sử dụng mã hóa base64 để nhận được 1 đoạn mã hóa.
  + Signature : Là phần kết hợp giữa header và payload đã được mã hóa base64, sau đó nó sẽ kết hợp với 1 cái secretkey và thuật toán mã hóa ở phần header để mã hóa thu được signature cuối cùng. Đây cũng là phần quan trọng nhất để xác định rằng accessToken không bị thay đổi khi truyền tin.
* Sau khi có được 3 thành phần trên, kết hợp chúng và phân tách mỗi phần bằng dấu chấm.



1. Đăng ký

**Scenario chuẩn :**

1. Người dùng chọn đăng ký tài khoản
2. Login Page hiển thị form chọn role
3. Người dùng chọn role muốn đăng ký
4. Login Page gọi tới form SignUp Page
5. SignUp Page hiển thị form cho người dùng
6. Người dùng nhập thông tin cần đăng ký vào form SignUp
7. SignUp Page gọi tới UserService yêu cầu xử lý
8. UserService gọi xuống Database User để yêu cầu kiểm tra
9. Database User kiểm tra các thông tin đăng ký
10. Trả về thành công cho UserService
11. UserService thông báo thành công cho SignUp Page
12. SignUp Page điều hướng người dùng về LoginPage
13. Login Page hiển thị cho người dùng

**Scenario ngoại lệ :**

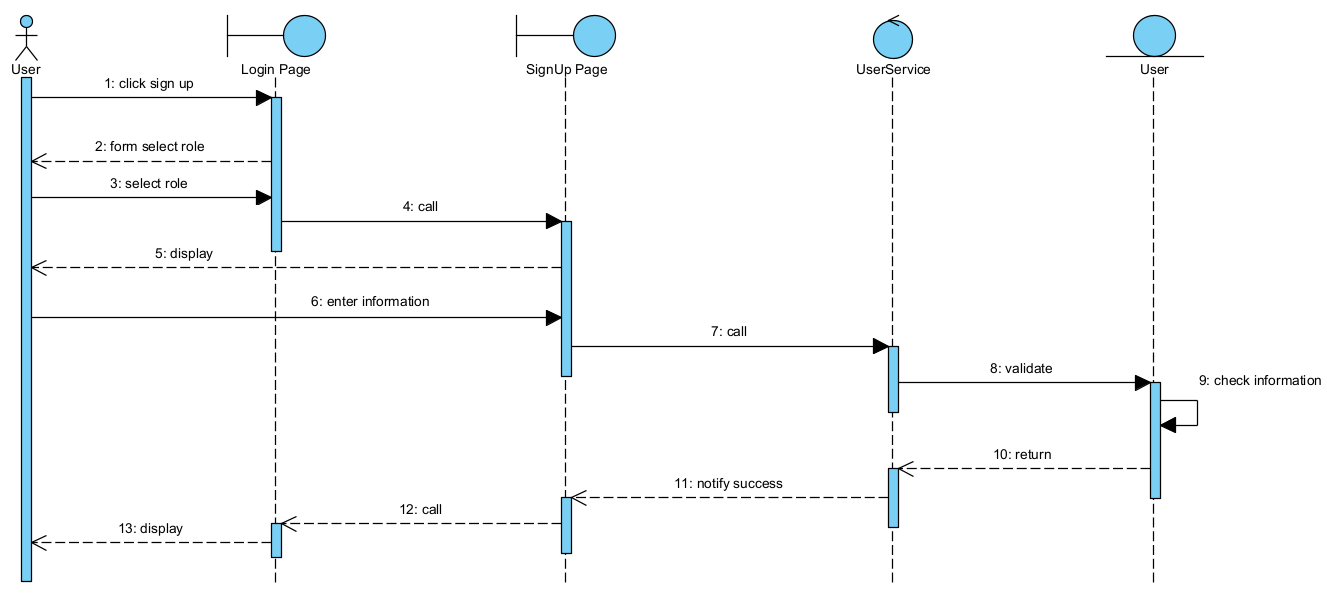
6.1. Người dùng không nhập thông tin mà chọn đăng ký

6.1.1. Hệ thống yêu cầu người dùng nhập đủ thông tin

9.1. Người dùng nhập username, email, phoneNumber đã tồn tại

9.1.1. Hệ thống thông báo tên username, email, phoneNumber đã tồn tại

* Dưới đây là sơ đồ luồng hoạt động của việc đăng ký tài khoản :



* **API :**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Request** | | | | | | |
| **Mô tả** | Gửi thông tin đăng kí tài khoản mới | | | | | |
| **Enpoint** | /user/register | | | | | |
| **Method** | POST | | | | | |
| **Param** | Không | | | | | |
| **Query** | Không | | | | | |
| **Body** | **Kiểu** | JSON | | | | |
| **Dữ liệu** | **Tên trường** | | **Kiểu** | **Mô tả** | |
| username | | string | Tên tài khoản | |
| password | | string | Mật khẩu | |
| email | | string | Email | |
| phoneNumber | | string | Số điện thoại | |
| fullName | | string | Tên đầy đủ | |
| Role | | string | Role | |
| address | | string | Địa chỉ | |
| **Response** | | | | | | |
| **Kiểu** | Object | | | | | |
| **Định dạng dữ liệu** | **Tên trường** | | **Kiểu** | | | **Mô tả** |
| accessToken | | string | | | Token của user |
| Role | | string | | | Role |
| **VD** |  | | | | | |

1. Đăng nhập sử dụng username/password

**Scenario chuẩn :**

1. Người dùng nhập thông tin đăng nhập vào Login Form
2. Login Form gọi tới UserService để kiểm tra thông tin
3. UserService gọi tới database User để kiểm tra thông tin tài khoản, mật khẩu
4. Database kiểm tra xong trả thông báo về cho UserService
5. UserService trả về accessToken + role cho Login Page
6. Login Page gọi đến Home Page ứng với từng role
7. Home Page hiển thị cho người dùng

**Scenario ngoại lệ :**

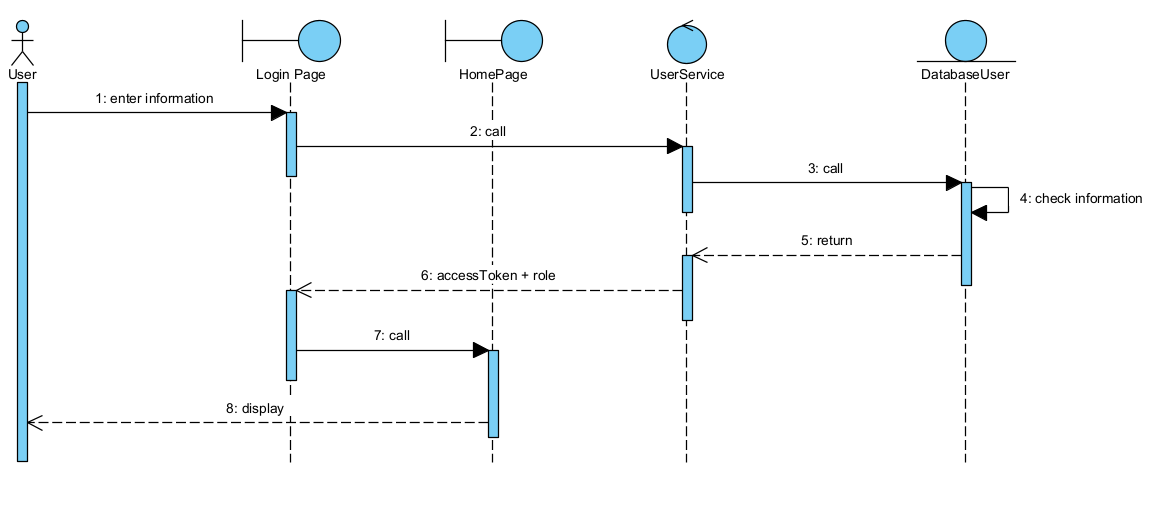
1.1. Người dùng không nhập thông tin mà chọn đăng nhập

1.1.1.Hệ thống thông báo yêu cầu người dùng nhập đầy đủ thông tin

4.1. Database User kiểm tra username hoặc password không đúng

4.1.1. Hệ thống thông báo tên tài khoản hoặc mật khẩu không chính xác

* Dưới đây là luồng hoạt động của chức năng đăng nhập :



* **API :**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Request** | | | | | | |
| **Mô tả** | Gửi thông tin đăng nhập | | | | | |
| **Enpoint** | /user/login | | | | | |
| **Method** | POST | | | | | |
| **Param** | Không | | | | | |
| **Query** | Không | | | | | |
| **Body** | **Kiểu** | JSON | | | | |
| **Dữ liệu** | **Tên trường** | | **Kiểu** | **Mô tả** | |
| Username | | string | Email của user | |
| password | | string | Mật khẩu của user | |
| **Response** | | | | | | |
| **Kiểu** | Object | | | | | |
| **Định dạng dữ liệu** | **Tên trường** | | **Kiểu** | | | **Mô tả** |
| accessToken | | string | | | Token của user |
| Role | | string | | | role |
| **VD** |  | | | | | |

1. Đăng nhập bằng google

**Scenario chuẩn :**

1. Người dùng chọn đăng nhập bằng google
2. Login Page gọi tới Oauth2Service để yêu cầu Url chuyển hướng
3. Oauth2Service trả lại link cho Login Page
4. Login Page điều hướng tới trang đăng nhập google
5. Trang đăng nhập google hiển thị cho người dùng
6. Người dùng nhập thông tin tài khoản google
7. Google Page gọi tới OpenID Provider để xác minh
8. OpenID Provider xác minh tài khoản google
9. OpenID Provider gửi mã code về cho google Page
10. Google trả về cho Login Page 1 authorization code
11. Login Page chuyển code cho Oauth2Service
12. Oauth2Service gửi mã code lên cho OpenID Provider yêu cầu xác mình nhận ID token và accessToken
13. OpenID Provider xác minh mã code
14. OpenID Provider gửi Id token và accessToken cho Oauth2Service
15. Oauth2Service trả về accessToken và role cho Login Page
16. Login Page gọi tới Home Page
17. Home Page hiển thị thông tin theo role

**Scenario ngoại lệ :**

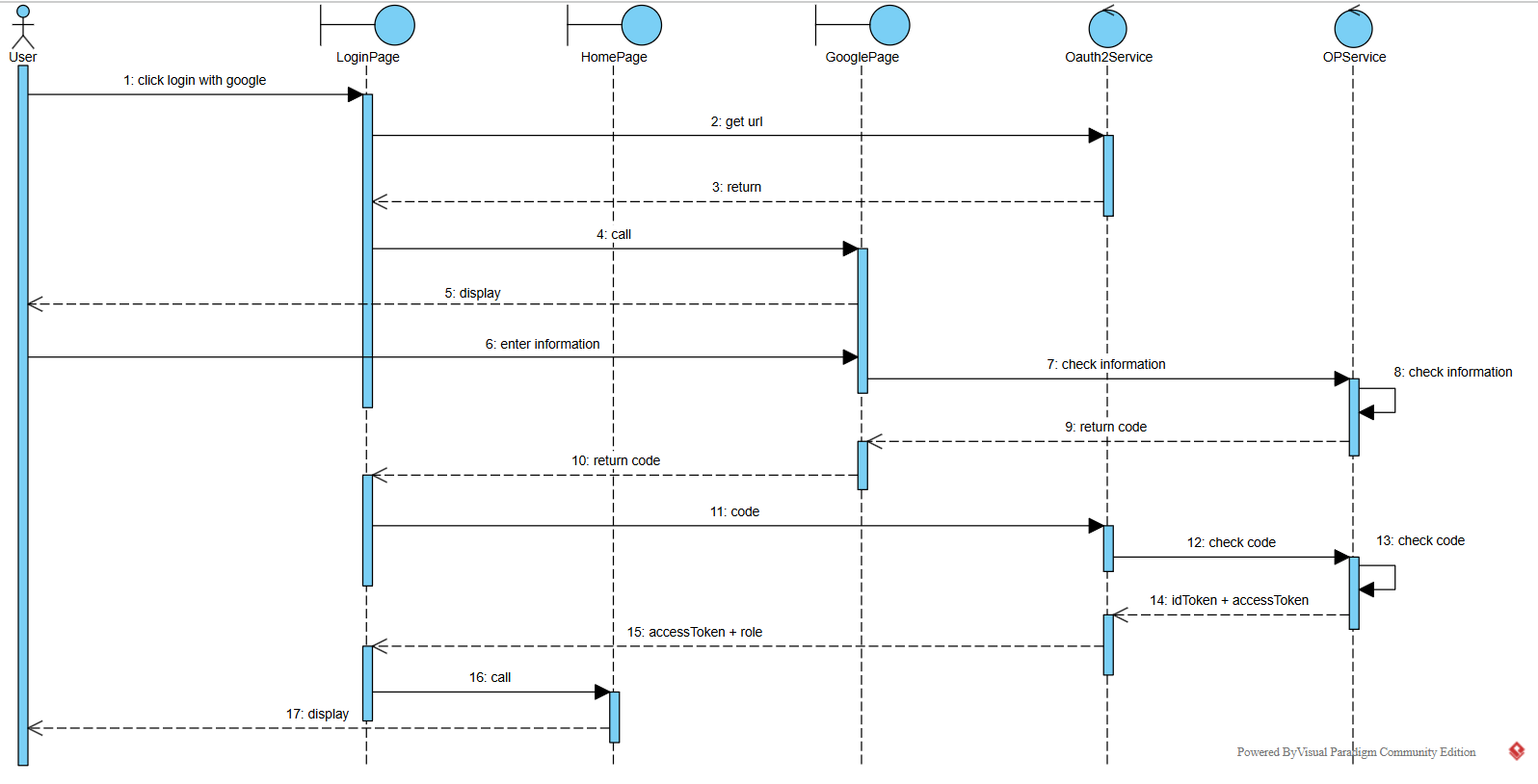
8.1 . OpenID Provider xác minh tài khoản không hợp lệ

8.1.1. Thông báo đăng nhập không thành công

12.1. OpenID Provider xác minh mã code không hợp lêk

12.1.1.Thông báo đăng nhập google không thành công

* Dưới đây là sơ đồ luồng hoạt động :



* API :

Lấy url điều hướng :

|  |  |
| --- | --- |
| **Request** | |
| **Mô tả** | Lấy url điều hướng tới đăng nhập google |
| **Enpoint** | /auth/google-url |
| **Method** | GET |
| **Param** | Không |
| **Query** | Không |
| **Body** | Không |
| **Response** | |
| **VD** |  |

Đăng nhập google :

|  |  |
| --- | --- |
| **Request** | |
| **Mô tả** | Đăng nhập google |
| **Enpoint** | /auth/logIn |
| **Method** | POST |
| **Param** | Không |
| **Query** | Không |
| **Body** | Không |
| **Response** | |
| **VD** |  |

1. Quên mật khẩu

**Scenario chuẩn :**

1. Người dùng chọn quên mật khẩu
2. Login Page hiển thị form yêu cầu nhập username
3. Người dùng nhập username và chọn gửi mã code tới email
4. Login Page gọi tới UserService yêu cầu gửi mã code tới email
5. UserService gọi tới Database User để kiểm tra
6. Database User kiểm tra username
7. Database trả email cho UserSerivce
8. UserService gửi mã code tới email Service
9. UserService trả về Login Page thông báo gửi mã code thành công
10. LoginPage hiển thị 1 form nhập code và mật khẩu mới
11. Người dùng nhập mã code và mật khẩu mới
12. Login Page gọi tới UserService yêu cầu đặt lại mật khẩu
13. UserService kiểm tra mã code
14. UserService gọi tới Databse User để update mật khẩu
15. Database User update mật khẩu người dùng
16. Database User thông báo thành công cho UserService
17. UserService thông báo thành công cho Login Page
18. Login Page hiển thị cho người dùng

**Scenario ngoại lệ :**

2.1.Người dùng không nhập username mà chọn lấy mã code luôn

2.1.1.Hệ thống thông báo yêu cầu nhập username

6.1. Database User kiểm tra username sai

6.1.1.Hệ thống thông báo username không tồn tại

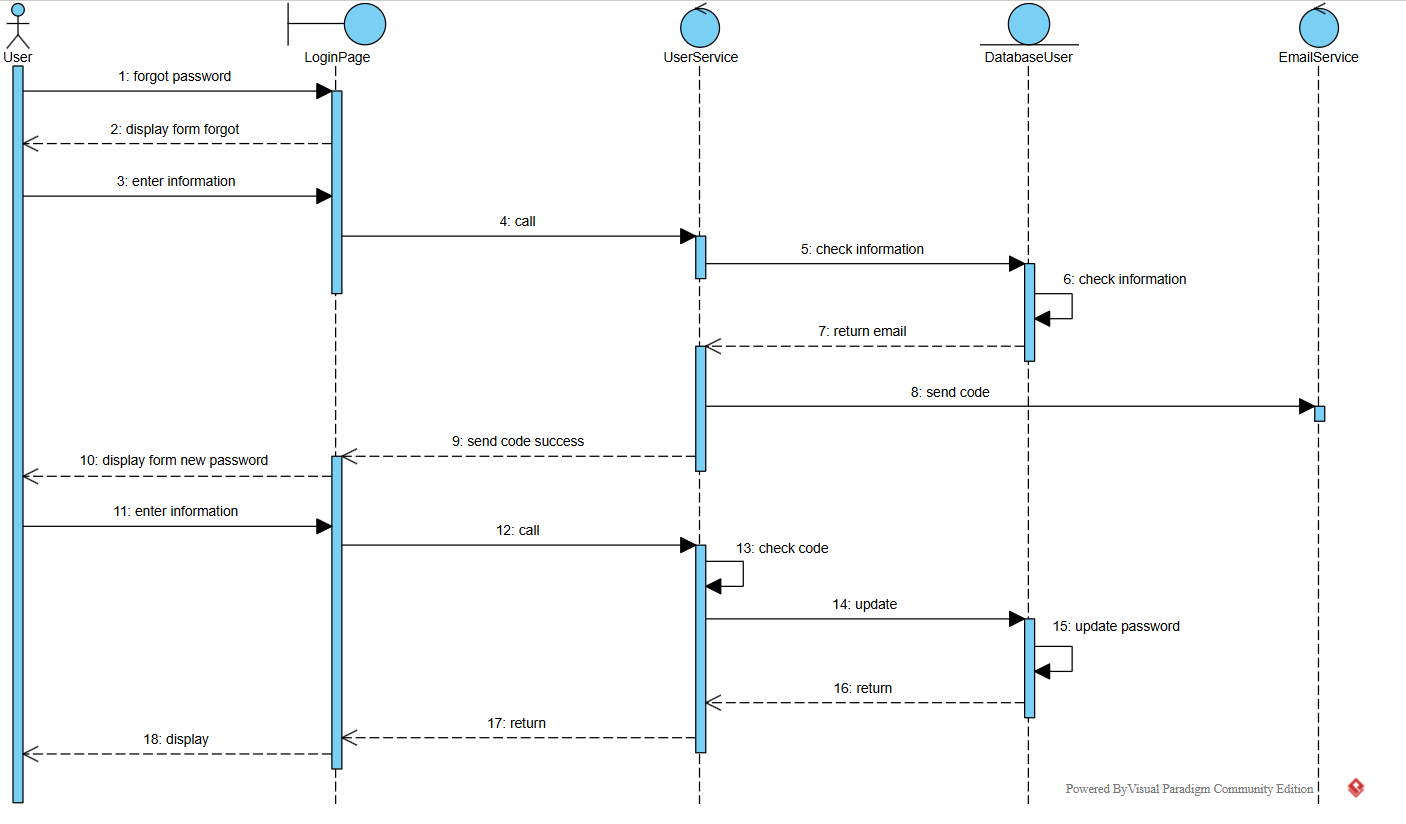
10.1. Người dùng không nhập mật khẩu mới, mã code

10.1.1.Hệ thống thông báo yêu cầu nhập mã code, mật khẩu mới

12.1. Người dùng nhập mã code không hợp lệ

12.1.1. Hệ thống thông báo mã code không hợp lệ

- Sơ đồ tuần tự :



* **API :**

**Gửi mã code qua email**

|  |  |
| --- | --- |
| **Request** | |
| **Mô tả** | Gửi mã xác thực qua email |
| **Enpoint** | /user/send-code-email |
| **Method** | POST |
| **Param** | Username |
| **Query** | Không |
| **Body** | Không |
| **Response** | |
| **VD** |  |

**Đặt lại mật khẩu :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request** | | | | |
| **Mô tả** | Đặt lại mật khẩu | | | |
| **Enpoint** | /user/recover-password | | | |
| **Method** | POST | | | |
| **Param** | Không | | | |
| **Query** | Không | | | |
| **Body** | **Kiểu** | JSON | | |
| **Dữ liệu** | **Tên trường** | **Kiểu** | **Mô tả** |
| newPassword | string | Mật khẩu mới |
| Code | string | Mã xác thực |
| **Response** | | | | |
| **VD** |  | | | |

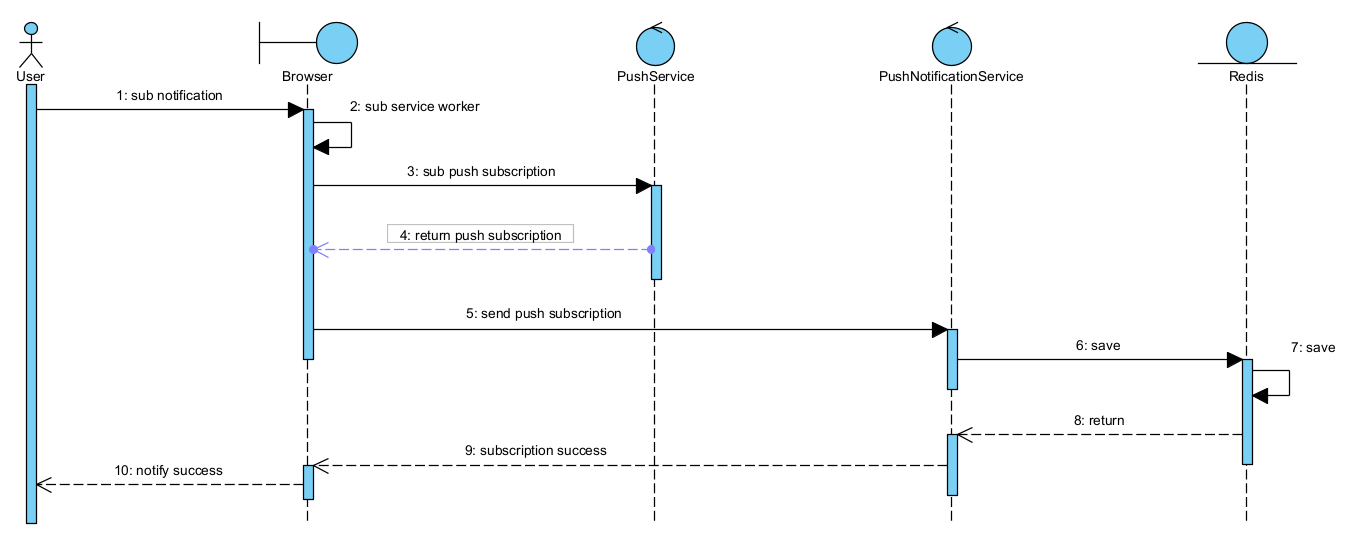
### Thông báo

1. Đăng ký nhận thông báo

**Scenario chuẩn :**

1. Người dùng chọn nhận thông báo đẩy
2. Browser thực hiện đăng ký Serivce Worker
3. Sau khi đăng ký Service Worker, Browser đăng ký Push Subscription với Push Service
4. Push Service trả về Subscription với url nhận thông báo, p256dh và auth secret
5. Browser gửi push subcription xuống PushNotificationService
6. PushNotificationService gọi tới Redis để lưu trữ thông tin
7. Redis lưu lại Push Subscription
8. Redis trả về lưu trữ thành công cho PushNotificationSerivice
9. PushNotificationService thông báo đăng ký thành công
10. Browser thông báo cho người dùng

* Sơ đồ tuần tự :



* API :

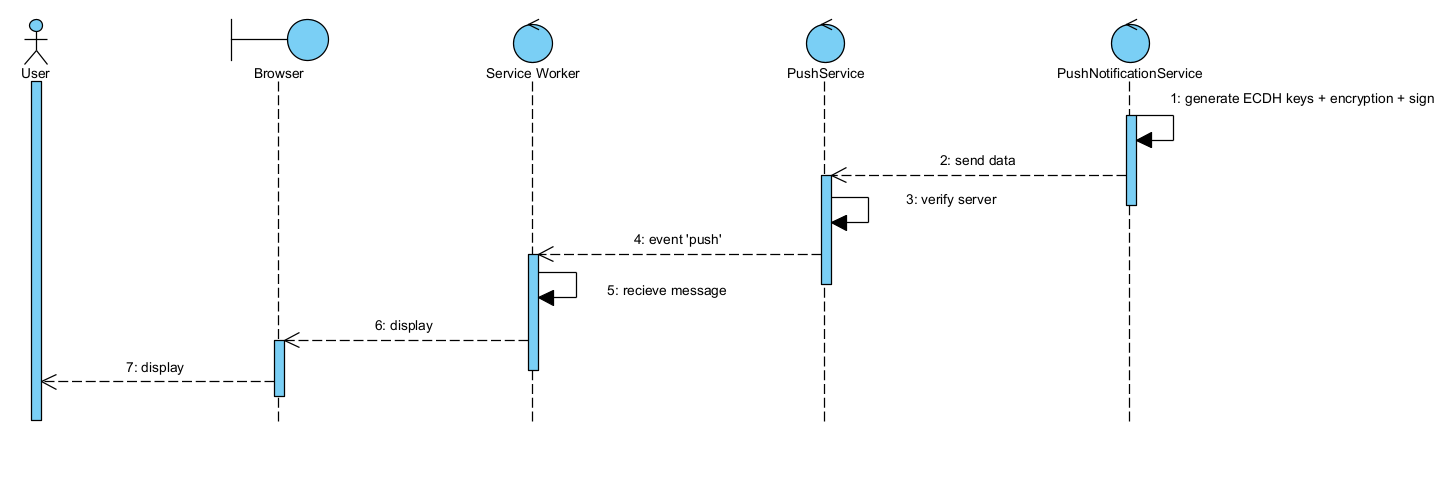
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request** | | | | |
| **Mô tả** | Đăng ký nhận thông báo. | | | |
| **Enpoint** | /push-notification/subscribe | | | |
| **Method** | POST | | | |
| **Param** | Không | | | |
| **Query** | Không | | | |
| **Body** | **Kiểu** | JSON | | |
| **Dữ liệu** | **Tên trường** | **Kiểu** | **Mô tả** |
| endpoint | string | Endpoint gửi tới push service |
| expirationTime | Date | Ngày hết hạn |
| P256dh | string | Publickey client |
| auth | String | Chuỗi định danh |
| **Response** | | | | |
| **VD** |  | | | |

1. Thông báo đẩy tới người dùng

**Scenario chuẩn :**

1. Push notification sinh 1 cặp ECDH key và mã hóa message với shared key(public key client, private key server), auth, salt, sau đó ký với private key server
2. PushNotificationService gửi tin nhắn lên Push Service
3. Push Service định danh Server bằng cách giải mã chữ ký trước đó
4. Push Service dựa vào url để đẩy thông báo về Service Worker
5. Service Worker nhận được sự kiện ‘push’
6. Serivce Worker hiển thị cho Browser
7. Browser hiển thị cho người dùng

* Sơ đồ tuần tự :

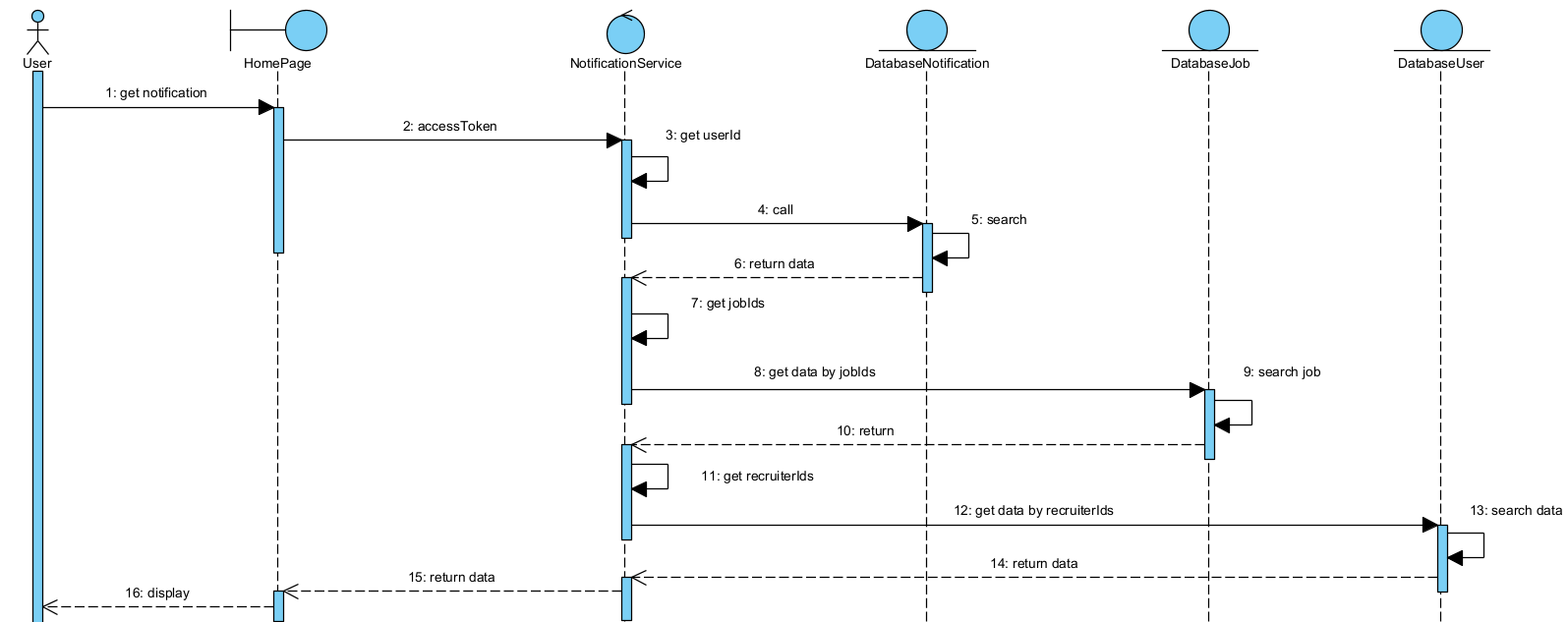


1. Xem thông báo

**Scenario chuẩn :**

1. Người dùng nhấn vào phần xem thông báo trên giao diện trang chủ.
2. Trang chủ lấy accessToken của người dùng và gửi đến NotificationService.
3. NotificationService giải mã accessToken để lấy userId.
4. NotificationService gửi yêu cầu tới DatabaseNotification để lấy thông báo của user theo userId.
5. DatabaseNotification thực hiện tìm kiếm các thông báo tương ứng.
6. DatabaseNotification trả về danh sách thông báo.
7. NotificationService lấy ra các jobId từ danh sách thông báo.
8. NotificationService gửi yêu cầu đến DatabaseJob để lấy thông tin chi tiết các công việc theo jobId.
9. DatabaseJob thực hiện tìm kiếm thông tin các công việc.
10. DatabaseJob trả về thông tin các công việc.
11. NotificationService lấy recruiterId từ thông tin các công việc đã nhận được.
12. NotificationService gửi yêu cầu đến DatabaseUser để lấy thông tin recruiter theo recruiterId.
13. DatabaseUser tìm kiếm thông tin người tuyển dụng theo recruiterId.
14. DatabaseUser trả về thông tin recruiter cho NotificationService.
15. NotificationService trả toàn bộ dữ liệu về cho trang chủ.
16. Trang chủ hiển thị danh sách thông báo với thông tin công việc và nhà tuyển dụng cho người dùng.

* Sơ đồ tuần tự :



* API :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Request** | | | | | |
| **Mô tả** | Xem thông báo | | | | |
| **Enpoint** | /notifications | | | | |
| **Method** | GET | | | | |
| **Param** | **Tên trường** | **Kiểu dữ liệu** | | **Mô tả** | |
| Page | Int | | Trang | |
| Size | Int | | Kích thước trang | |
| **Query** | Không | | | | |
| **Body** | Không | | | | |
| **Response** | | | | | |
| **Kiểu** | Page<Object> | | | | |
| **Định dạng kiểu dữ liệu** | **Tên trường** | | **Kiểu dữ liệu** | | **Mô tả** |
| interactId | | Long | | Người tương tác |
| jobId | | Long | | Id công việc |
| fullName | | String | | Tên người tương tác |
| jobName | | String | | Tên công việc |
| Type | | String | | Loại thông báo |
| hasSeen | | Boolean | | Đã xem chưa |
| **VD** |  | | | | |

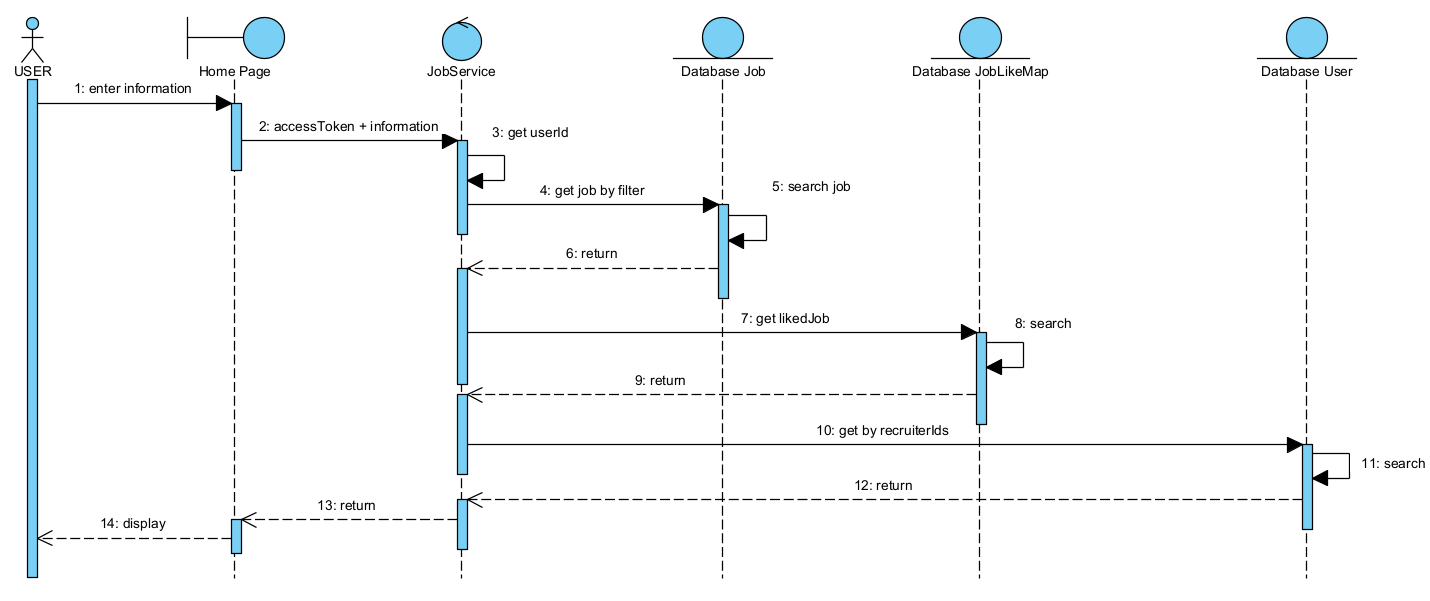
### Người tìm việc

1. Tìm kiếm công việc

**Scenario chuẩn :**

1. Người dùng truy cập vào trang chủ và nhập thông tin tìm kiếm công việc.
2. Trang chủ gửi accessToken cùng với thông tin tìm kiếm đến JobService.
3. JobService sử dụng accessToken để lấy userId của người dùng.
4. JobService gửi yêu cầu tìm kiếm công việc theo bộ lọc đến cơ sở dữ liệu Job.
5. Cơ sở dữ liệu Job thực hiện tìm kiếm các công việc dựa trên bộ lọc đã nhận được.
6. Cơ sở dữ liệu Job trả về danh sách các công việc phù hợp.
7. JobService gửi yêu cầu đến cơ sở dữ liệu JobLikeMap để lấy danh sách các công việc mà người dùng đã thích.
8. Cơ sở dữ liệu JobLikeMap tìm kiếm các công việc đã được người dùng đánh dấu yêu thích.
9. Cơ sở dữ liệu JobLikeMap trả về danh sách các công việc đã thích cho JobService.
10. JobService lấy danh sách recruiterIds từ các công việc và gửi yêu cầu đến cơ sở dữ liệu User để lấy thông tin nhà tuyển dụng.
11. Cơ sở dữ liệu User tìm kiếm thông tin của các nhà tuyển dụng theo recruiterIds.
12. Cơ sở dữ liệu User trả về thông tin chi tiết của các nhà tuyển dụng.
13. JobService tổng hợp toàn bộ dữ liệu và trả kết quả về cho Home Page.
14. Trang chủ hiển thị danh sách công việc, bao gồm cả các công việc đã thích và thông tin nhà tuyển dụng, cho người dùng.

* Dưới đây là sơ đồ tuần tự :



* **API :**

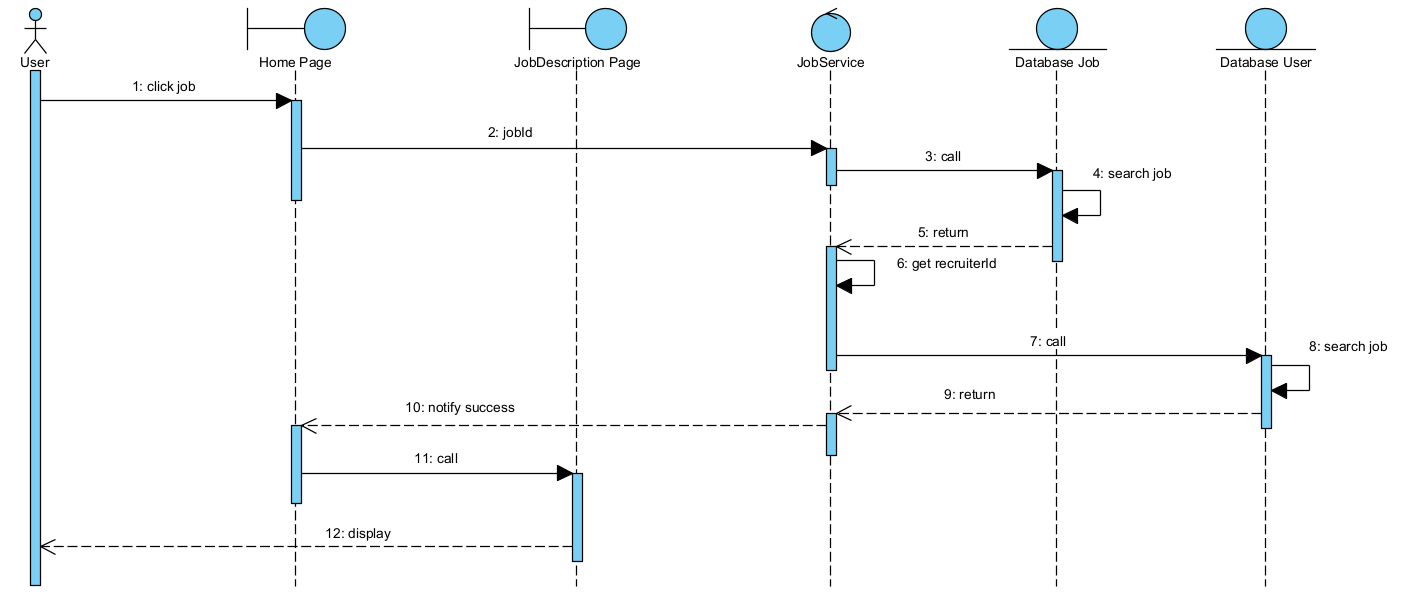
|  |  |  |  |
| --- | --- | --- | --- |
| **Request** | | | |
| **Mô tả** | Tìm kiếm công việc | | |
| **Enpoint** | /job/filter | | |
| **Method** | GET | | |
| **Param** | **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| jobName | String | Tên công việc |
| Occupation | String | Lĩnh vực |
| Province | String | Tỉnh thành |
| jobType | String | Hình thức làm việc |
| jobLevel | String | Cấp bậc |
| educationLevel | String | Trình độ học vấn |
| experience | String | Kinh nghiệm làm việc |
| minSalary | String | Lương tối thiểu |
| maxSalary | String | Lương cao nhất |
| page | Int | Trang |
| pize | Int | Kích thước trang |
| **Query** | Không | | |
| **Body** | Không | | |
| **Response** | | | |
| **Kiểu** | Object | | |
| **VD** |  | | |

1. Xem chi tiết công việc

**Scenario chuẩn :**

1. Người dùng nhấn vào một công việc trên trang chủ (home page).
2. Trang chủ gửi jobId đến trang mô tả công việc (job description page).
3. Job description page gọi đến jobService để lấy thông tin công việc tương ứng với jobId.
4. JobService gửi yêu cầu truy vấn đến database job để tìm công việc theo jobId.
5. Database job trả về thông tin chi tiết của công việc.
6. JobService lấy recruiterId từ thông tin công việc vừa nhận được.
7. JobService tiếp tục gọi đến database user để tìm thông tin của recruiter theo recruiterId.
8. Database user tìm kiếm thông tin recruiter trong bảng người dùng.
9. Database user trả về thông tin recruiter cho jobService.
10. JobService thông báo kết quả thành công về cho trang job description page.
11. Job description page gửi lệnh cho home page để hiển thị thông tin chi tiết công việc.
12. Home page hiển thị thông tin chi tiết công việc cho người dùng.

* Sơ đồ tuẩn tự :



* **API :**

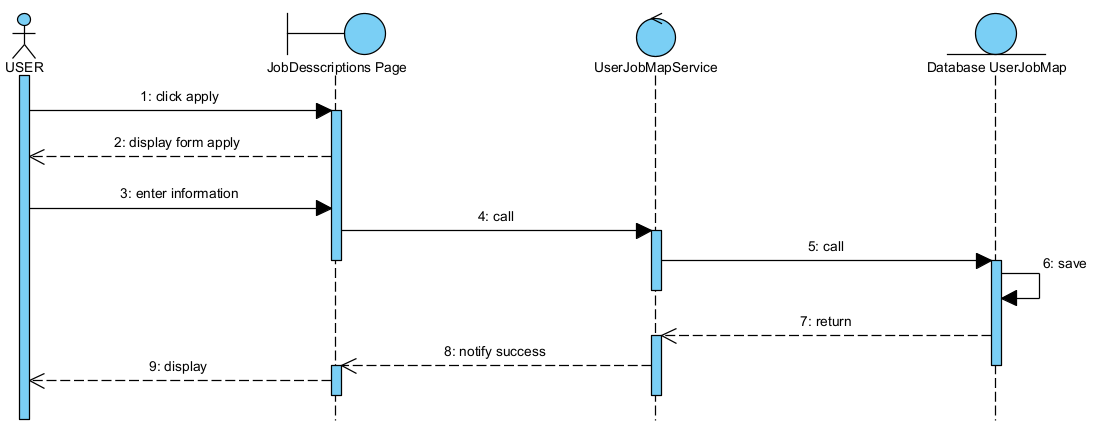
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Request** | | | | | |
| **Mô tả** | Lấy thông tin chi tiết công việc | | | | |
| **Enpoint** | /job/descriptions | | | | |
| **Method** | GET | | | | |
| **Param** | **Tên trường** | **Kiểu dữ liệu** | | **Mô tả** | |
| jodId | Long | | Id công việc | |
| **Query** | Không | | | | |
| **Body** | Không | | | | |
| **Response** | | | | | |
| **Kiểu** | Object | | | | |
| **Định dạng dữ liệu** | **Tên trường** | | **Kiểu dữ liệu** | | **Mô tả** |
| id | | Long | | Id công việc |
| UserOutput | | Object | | Nhà tuyển dụng |
| jobName | | String | | Tên công việc |
| minSalary | | Double | | Lương thấp nhất |
| maxSalary | | Double | | Lương cao nhất |
| Address | | String | | Địa chỉ |
| expirationDate | | String | | Ngày hết hạn |
| occupationName | | String | | Lĩnh vực |
| Experience | | String | | Kinh nghiệm |
| headCount | | Int | | Số lượng tuyển |
| jobType | | String | | Hình thức làm việc |
| JobLevel | | String | | Cấp bậc |
| educationLevel | | String | | Trình độ học vấn |
| Descriptions | | String | | Mô tả công việc |
| requiredJobList | | String | | Yêu cầu công việc |
| employeeBenefitList | | String | | Lợi ích công việc |
| hasLiked | | Boolean | | Công việc được thích |
| **VD** |  | | | | |

1. Nộp đơn ứng tuyển

**Scenario chuẩn :**

1. Người dùng nhấn vào nút "Apply" trên trang mô tả công việc.
2. Trang JobDescriptions Page hiển thị form ứng tuyển để người dùng điền thông tin.
3. Người dùng điền các thông tin cần thiết vào form (họ tên, email, CV, v.v.).
4. JobDescriptions Page gọi đến UserJobMapService để xử lý thông tin ứng tuyển.
5. UserJobMapService gửi yêu cầu lưu thông tin đến Database UserJobMap.
6. Database UserJobMap lưu thông tin ứng tuyển vào bảng dữ liệu.
7. Database phản hồi kết quả lưu thành công về cho UserJobMapService.
8. UserJobMapService thông báo lại thành công cho JobDescriptions Page.
9. JobDescriptions Page hiển thị thông báo ứng tuyển thành công cho người dùng.

* Dưới đây là sơ đồ tuần tự :



* **API :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request** | | | | |
| **Mô tả** | Nộp đơn ứng tuyển | | | |
| **Enpoint** | /interact/apply | | | |
| **Method** | POST | | | |
| **Param** | Không | | | |
| **Query** | Không | | | |
| **Body** | **Kiểu** | Form-data | | |
| **Dữ liệu** | **Tên trường** | **Kiểu** | **Mô tả** |
| applyJobInputString | String | Chứa jobId và thông tin giới thiệu bản thân |
| filePDF | File | File CV |
| **Response** | | | | |
| **VD** |  | | | |

1. Xem thông tin cá nhân và thay đổi thông tin

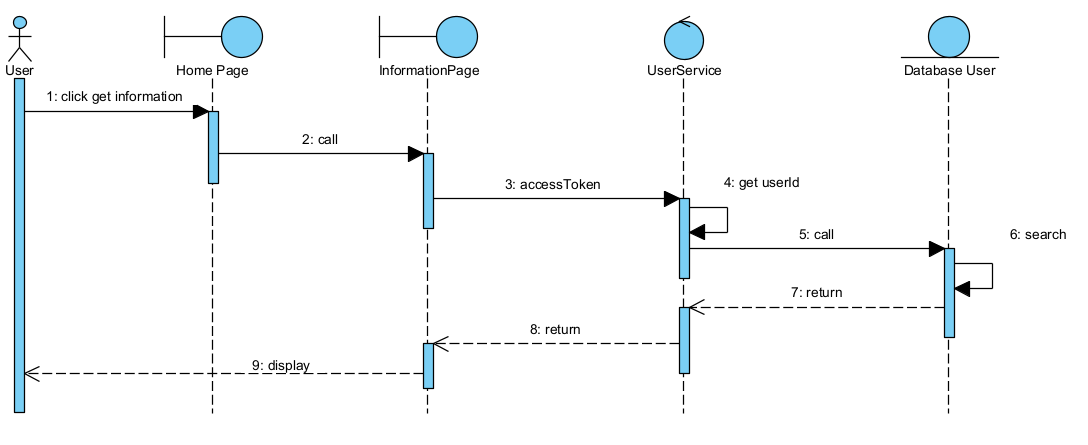
**Scenario xem thông tin chuẩn :**

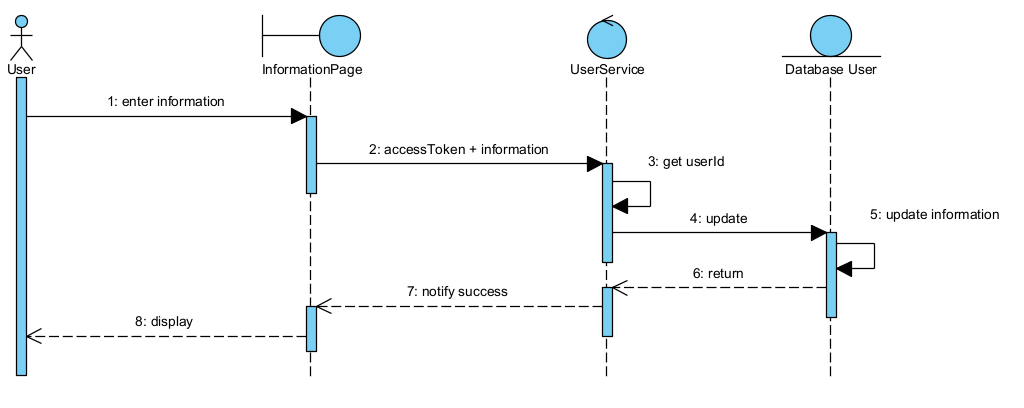
1. Người dùng nhấn vào nút "Get Information" trên giao diện trang chủ.
2. Home Page gửi yêu cầu đến trang InformationPage để xử lý.
3. InformationPage lấy access token từ người dùng để xác thực.
4. InformationPage gửi access token đến UserService để lấy userId tương ứng.
5. UserService gọi đến Database User để tìm kiếm thông tin người dùng dựa trên userId.
6. Database User thực hiện truy vấn tìm kiếm thông tin.
7. Database trả về thông tin người dùng cho UserService.
8. UserService trả kết quả về cho InformationPage.
9. InformationPage hiển thị thông tin người dùng cho người dùng.

**Scenario chỉnh sửa thông tin :**

1. Người dùng nhập thông tin cần cập nhật trên giao diện trang InformationPage.
2. InformationPage gửi accessToken và thông tin mới đến UserService.
3. UserService lấy userId từ accessToken.
4. UserService gửi yêu cầu cập nhật thông tin đến Database User.
5. Database User cập nhật thông tin người dùng trong cơ sở dữ liệu.
6. Database phản hồi kết quả cập nhật về cho UserService.
7. UserService thông báo cập nhật thành công về cho InformationPage.
8. InformationPage hiển thị thông báo thành công cho người dùng.

* Dưới đây là sơ đồ tuần tự :





* **API xem thông tin :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Request** | | | |
| **Mô tả** | Xem thông tin cá nhân | | |
| **Enpoint** | /user | | |
| **Method** | GET | | |
| **Param** | Không | | |
| **Query** | Không | | |
| **Body** | Không | | |
| **Response** | | | |
| **Kiểu** | Object | | |
| **Định dạng kiểu dữ liệu** | **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| fullName | String | Tên người dùng |
| imageUrl | String | Hình ảnh |
| backgroundImage | String | Hình nền |
| Description | String | Mô tả |
| Birthday | DateTime | Ngày sinh |
| Address | String | Địa chỉ |
| phoneNumber | String | Số điện thoại |
| Email | String | Email |
| Role | String | Role |
| **VD** |  | | |

* **API chỉnh sửa thông tin :**

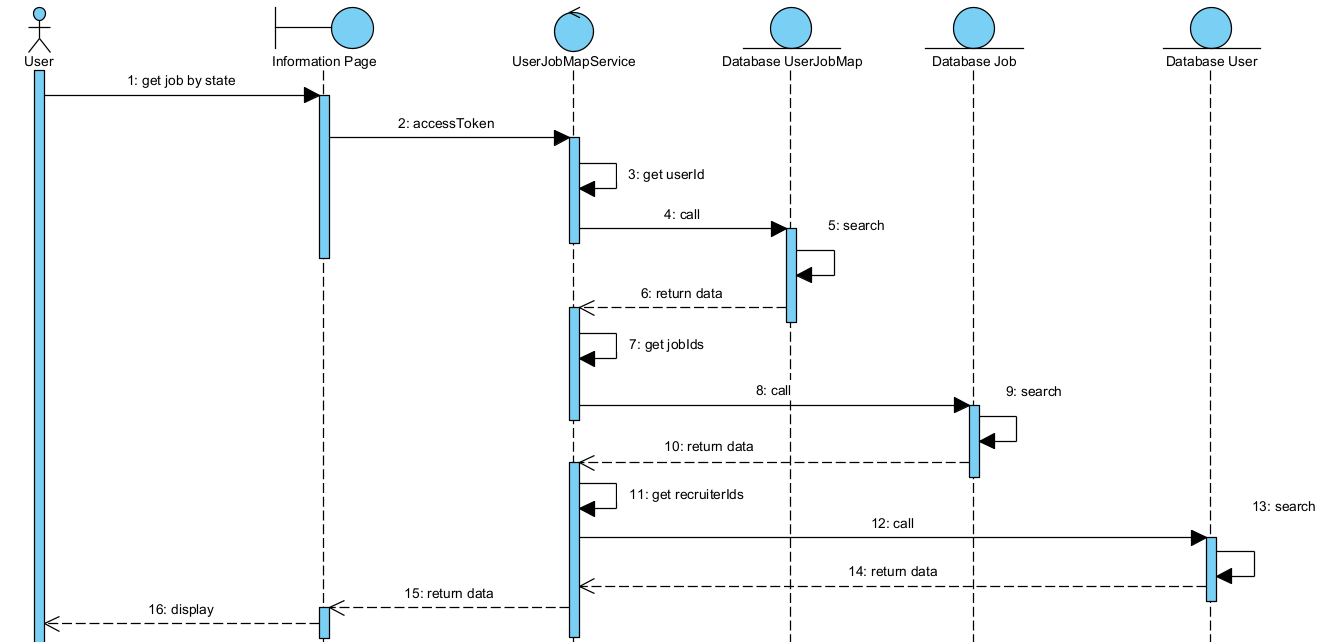
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request** | | | | |
| **Mô tả** | Thay đổi thông tin cá nhân | | | |
| **Enpoint** | /user/change-user-information | | | |
| **Method** | POST | | | |
| **Param** | Không | | | |
| **Query** | Không | | | |
| **Body** | **Kiểu** | Form-data | | |
| **Dữ liệu** | **Tên trường** | **Kiểu** | **Mô tả** |
| new\_user\_info | Tex | Endpoint gửi tới push service |
| Image | File | Ảnh đại diện |
| Background\_img | File | Ảnh nền |
| **Response** | | | | |
| **VD** |  | | | |

1. Xem danh sách công việc theo trạng thái

**Scenario chuẩn :**

1. Người dùng gửi yêu cầu xem danh sách công việc theo trạng thái từ giao diện Information Page.
2. Information Page gửi accessToken đến UserJobMapService để xác thực.
3. UserJobMapService lấy userId từ accessToken.
4. UserJobMapService gọi tới Database UserJobMap để lấy dữ liệu liên kết giữa user và công việc.
5. Database UserJobMap thực hiện tìm kiếm các bản ghi theo userId.
6. Database trả dữ liệu liên kết về cho UserJobMapService.
7. UserJobMapService trích xuất danh sách jobIds từ dữ liệu vừa nhận.
8. UserJobMapService gọi đến Database Job để lấy thông tin chi tiết các công việc.
9. Database Job thực hiện tìm kiếm các công việc theo danh sách jobIds.
10. Database Job trả về thông tin công việc cho UserJobMapService.
11. UserJobMapService tiếp tục lấy danh sách recruiterIds liên quan đến các công việc.
12. UserJobMapService gọi đến Database User để lấy thông tin nhà tuyển dụng.
13. Database User tìm kiếm thông tin các recruiter theo recruiterIds.
14. Database trả về dữ liệu recruiter cho UserJobMapService.
15. UserJobMapService trả toàn bộ thông tin công việc và nhà tuyển dụng về cho Information Page.
16. Information Page hiển thị kết quả cho người dùng.

* Sơ đồ tuần tự :



* **API :**

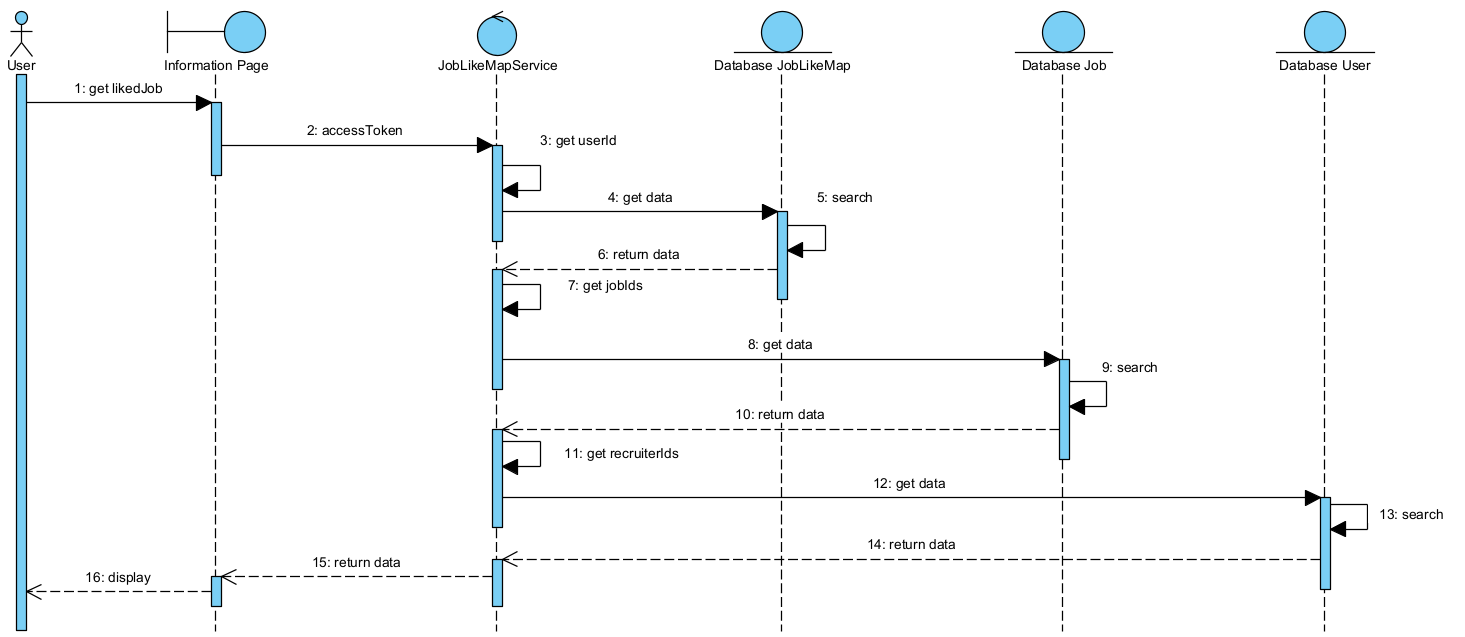
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Request** | | | | | |
| **Mô tả** | Lấy thông tin chi tiết công việc | | | | |
| **Enpoint** | /job/state | | | | |
| **Method** | GET | | | | |
| **Param** | **Tên trường** | **Kiểu dữ liệu** | | **Mô tả** | |
| state | String | | Trạng thái đơn ứng tuyển | |
| page | Int | | Trang | |
| size | Int | | Kích thước trang | |
| **Query** | Không | | | | |
| **Body** | Không | | | | |
| **Response** | | | | | |
| **Kiểu** | Page<Object> | | | | |
| **Định dạng dữ liệu** | **Tên trường** | | **Kiểu dữ liệu** | | **Mô tả** |
| id | | Long | | Id công việc |
| nameRecrutier | | String | | Tên nhà tuyển dụng |
| jobName | | String | | Tên công việc |
| minSalary | | Double | | Lương thấp nhất |
| maxSalary | | Double | | Lương cao nhất |
| Address | | String | | Địa chỉ |
| expirationDate | | String | | Ngày hết hạn |
| hasLiked | | Boolean | | Công việc được thích |
| imageUrl | | String | | Hình ảnh công ty |
| **VD** |  | | | | |

1. Xem công việc đã thích

**Scenario chuẩn :**

1. Người dùng gửi yêu cầu xem danh sách công việc đã thích từ giao diện Information Page.
2. Information Page gửi accessToken đến JobLikeMapService để xác thực.
3. JobLikeMapService giải mã accessToken để lấy userId.
4. JobLikeMapService gọi tới Database JobLikeMap để lấy dữ liệu các công việc mà user đã thích.
5. Database JobLikeMap tìm kiếm và trả kết quả phù hợp với userId.
6. Dữ liệu được trả về cho JobLikeMapService.
7. JobLikeMapService trích xuất danh sách jobIds từ dữ liệu nhận được.
8. JobLikeMapService gọi đến Database Job để lấy thông tin chi tiết các công việc.
9. Database Job thực hiện tìm kiếm theo danh sách jobIds.
10. Thông tin các công việc được trả về cho JobLikeMapService.
11. JobLikeMapService tiếp tục lấy danh sách recruiterIds liên quan đến các công việc đó.
12. JobLikeMapService gọi đến Database User để lấy thông tin nhà tuyển dụng.
13. Database User thực hiện tìm kiếm theo danh sách recruiterIds.
14. Database User trả về thông tin nhà tuyển dụng cho JobLikeMapService.
15. JobLikeMapService trả dữ liệu tổng hợp (công việc + nhà tuyển dụng) về cho Information Page.
16. Information Page hiển thị danh sách công việc đã thích cho người dùng.

* Sơ đồ tuần tự :



* **API :**

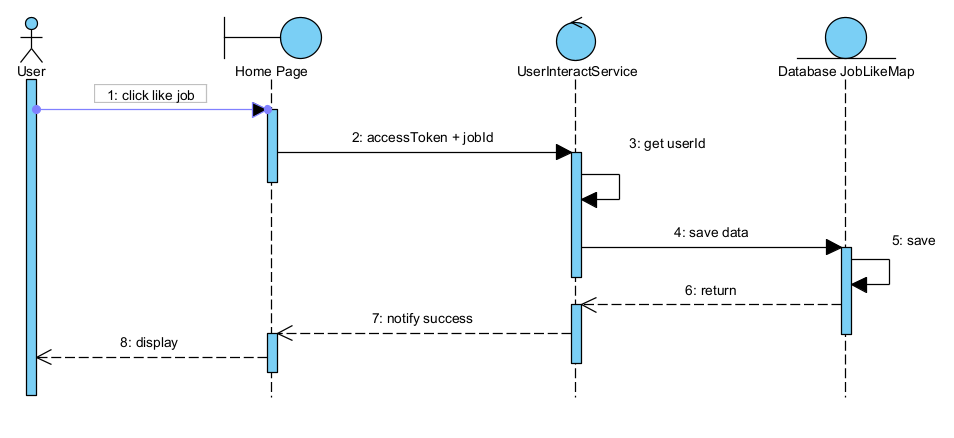
|  |  |  |  |
| --- | --- | --- | --- |
| **Request** | | | |
| **Mô tả** | Lấy thông tin chi tiết công việc | | |
| **Enpoint** | /job/liked | | |
| **Method** | GET | | |
| **Param** | Không | | |
| **Query** | Không | | |
| **Body** | Không | | |
| **Response** | | | |
| **Kiểu** | Page<Object> | | |
| **Định dạng dữ liệu** | **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| id | Long | Id công việc |
| nameRecrutier | String | Tên nhà tuyển dụng |
| jobName | String | Tên công việc |
| minSalary | Double | Lương thấp nhất |
| maxSalary | Double | Lương cao nhất |
| Address | String | Địa chỉ |
| expirationDate | String | Ngày hết hạn |
| hasLiked | Boolean | Công việc được thích |
| imageUrl | String | Hình ảnh công ty |
| **VD** |  | | |

1. Yêu thích công việc

**Scenario chuẩn :**

1. Người dùng nhấn nút "Like Job" trên giao diện **Home Page** để bày tỏ sự quan tâm đến một công việc.
2. Home Page gửi accessToken cùng với jobId đến **UserInteractService** để xử lý yêu cầu.
3. UserInteractService giải mã accessToken để lấy ra userId.
4. UserInteractService tạo dữ liệu tương ứng giữa user và job được thích, sau đó gửi yêu cầu lưu dữ liệu đến **Database JobLikeMap**.
5. Database JobLikeMap thực hiện thao tác lưu thông tin.
6. Database JobLikeMap trả về kết quả xác nhận lưu thành công.
7. UserInteractService thông báo kết quả thành công đến Home Page.
8. Home Page cập nhật lại giao diện và hiển thị thông báo cho người dùng.

* Sơ đồ tuần tự :



* API :

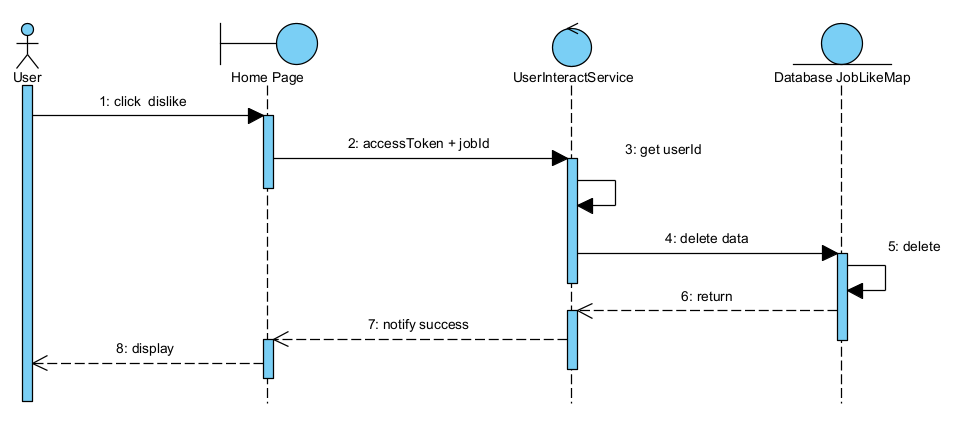
|  |  |  |  |
| --- | --- | --- | --- |
| **Request** | | | |
| **Mô tả** | Lấy thông tin chi tiết công việc | | |
| **Enpoint** | /interact/like | | |
| **Method** | POST | | |
| **Param** | **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| jodId | Long | Id công việc |
| **Query** | Không | | |
| **Body** | Không | | |
| **Response** | | | |
| **VD** |  | | |

1. Bỏ thích công việc

**Scenario chuẩn :**

1. Người dùng nhấn nút bỏ thích trên giao diện **Home Page** để xóa theo dõi công việc
2. Home Page gửi accessToken cùng với jobId đến **UserInteractService** để xử lý yêu cầu.
3. UserInteractService giải mã accessToken để lấy ra userId.
4. UserInteractService gửi yêu cầu xóa dữ liệu đến **Database JobLikeMap**.
5. Database JobLikeMap thực hiện thao tác xóa thông tin.
6. Database JobLikeMap trả về kết quả xác nhận xóa thành công.
7. UserInteractService thông báo kết quả thành công đến Home Page.
8. Home Page cập nhật lại giao diện và hiển thị thông báo cho người dùng.

* Sơ đồ tuần tự :



* API :

|  |  |  |  |
| --- | --- | --- | --- |
| **Request** | | | |
| **Mô tả** | Lấy thông tin chi tiết công việc | | |
| **Enpoint** | /interact/remove-like | | |
| **Method** | DELETE | | |
| **Param** | **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| jodId | Long | Id công việc |
| **Query** | Không | | |
| **Body** | Không | | |
| **Response** | | | |
| **VD** |  | | |

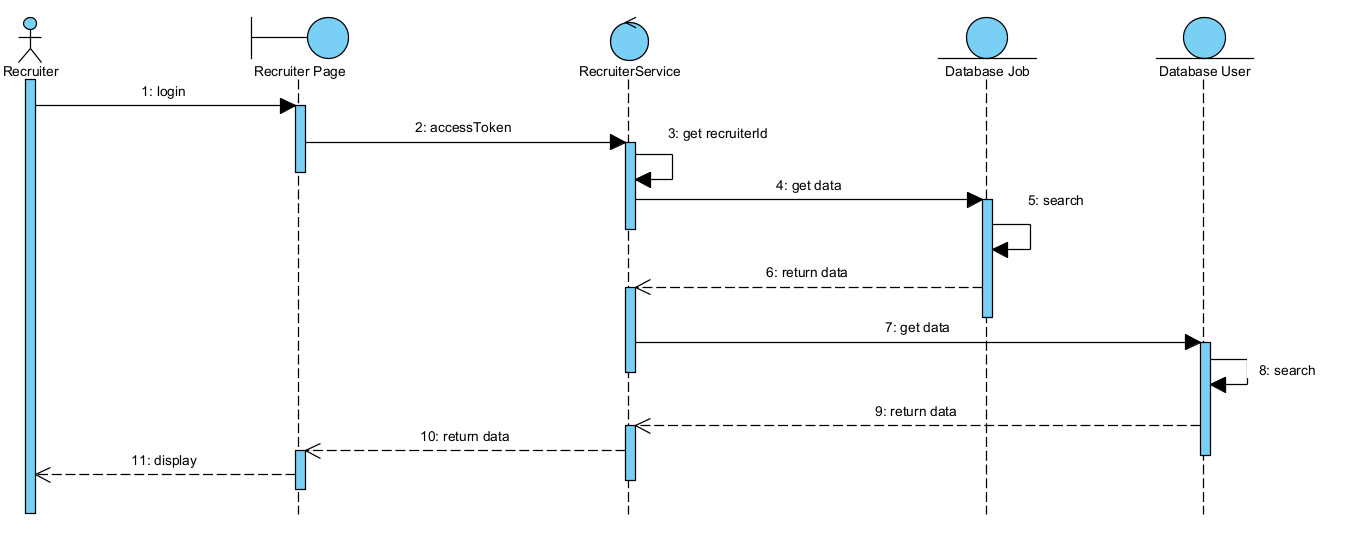
### Nhà tuyển dụng

1. Lấy ra thông tin và công việc đang tuyển

**Scenario chuẩn :**

1. Recruiter truy cập vào hệ thống và thực hiện hành động đăng nhập vào Recruiter Page.
2. Recruiter Page gửi accessToken đến RecruiterService để xác minh và lấy thông tin.
3. RecruiterService sử dụng accessToken để lấy recruiterId.
4. RecruiterService gửi yêu cầu lấy dữ liệu công việc từ Database Job.
5. Database Job thực hiện truy vấn và trả về dữ liệu công việc.
6. RecruiterService nhận dữ liệu từ Database Job.
7. RecruiterService tiếp tục gửi yêu cầu lấy dữ liệu người dùng đến Database User.
8. Database User thực hiện truy vấn và trả về dữ liệu người dùng.
9. RecruiterService nhận dữ liệu từ Database User.
10. RecruiterService gửi toàn bộ dữ liệu đã tổng hợp về cho Recruiter Page.
11. Recruiter Page hiển thị dữ liệu ra màn hình cho Recruiter.

* Sơ đồ tuần tự:



* API lấy thông tin :

|  |  |  |  |
| --- | --- | --- | --- |
| **Request** | | | |
| **Mô tả** | Xem thông tin cá nhân | | |
| **Enpoint** | /user | | |
| **Method** | GET | | |
| **Param** | Không | | |
| **Query** | Không | | |
| **Body** | Không | | |
| **Response** | | | |
| **Kiểu** | Object | | |
| **Định dạng kiểu dữ liệu** | **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| fullName | String | Tên người dùng |
| imageUrl | String | Hình ảnh |
| backgroundImage | String | Hình nền |
| Description | String | Mô tả |
| Birthday | DateTime | Ngày sinh |
| Address | String | Địa chỉ |
| phoneNumber | String | Số điện thoại |
| Email | String | Email |
| Role | String | Role |
| **VD** |  | | |

* API lấy công việc đang tuyển :

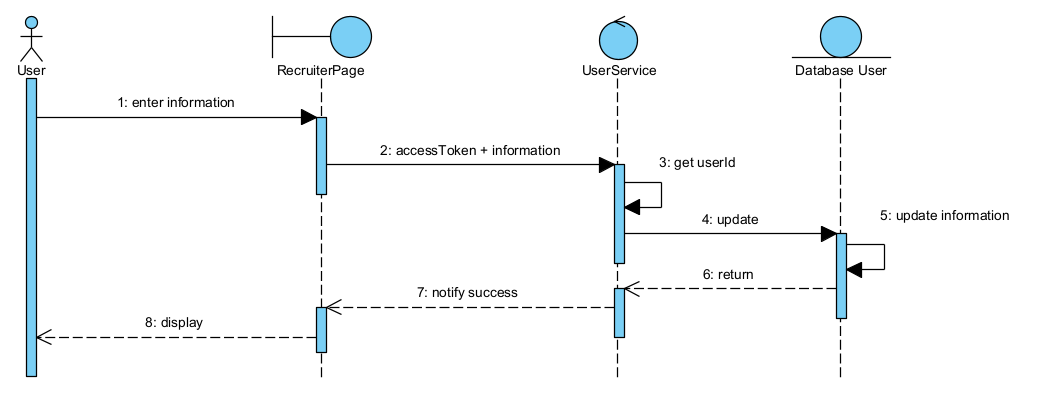
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Request** | | | | | |
| **Mô tả** | Lấy thông tin chi tiết công việc | | | | |
| **Enpoint** | /recruiter | | | | |
| **Method** | GET | | | | |
| **Param** | **Tên trường** | **Kiểu dữ liệu** | | **Mô tả** | |
| Page | Int | | Trang | |
| Size | Int | | Kích thước trang | |
| **Query** | Không | | | | |
| **Body** | Không | | | | |
| **Response** | | | | | |
| **Kiểu** | Page<Object> | | | | |
| **Định dạng dữ liệu** | **Tên trường** | | **Kiểu dữ liệu** | | **Mô tả** |
| id | | Long | | Id công việc |
| nameRecrutier | | String | | Tên nhà tuyển dụng |
| jobName | | String | | Tên công việc |
| minSalary | | Double | | Lương thấp nhất |
| maxSalary | | Double | | Lương cao nhất |
| Address | | String | | Địa chỉ |
| expirationDate | | String | | Ngày hết hạn |
| hasLiked | | Boolean | | Công việc được thích |
| imageUrl | | String | | Hình ảnh công ty |
| **VD** |  | | | | |

1. Thay đổi thông tin

**Scenario chuẩn :**

1. Người dùng nhập thông tin cần cập nhật trên giao diện trang RecruiterPage.
2. RecruiterPage gửi accessToken và thông tin mới đến UserService.
3. UserService lấy userId từ accessToken.
4. UserService gửi yêu cầu cập nhật thông tin đến Database User.
5. Database User cập nhật thông tin người dùng trong cơ sở dữ liệu.
6. Database phản hồi kết quả cập nhật về cho UserService.
7. UserService thông báo cập nhật thành công về cho RecruiterPage.
8. RecruiterPage hiển thị thông báo thành công cho người dùng.

* Sơ đồ tuần tự :



* API :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Request** | | | | |
| **Mô tả** | Thay đổi thông tin cá nhân | | | |
| **Enpoint** | /user/change-user-information | | | |
| **Method** | POST | | | |
| **Param** | Không | | | |
| **Query** | Không | | | |
| **Body** | **Kiểu** | Form-data | | |
| **Dữ liệu** | **Tên trường** | **Kiểu** | **Mô tả** |
| new\_user\_info | Tex | Endpoint gửi tới push service |
| Image | File | Ảnh đại diện |
| Background\_img | File | Ảnh nền |
| **Response** | | | | |
| **VD** |  | | | |

1. Thêm hoặc sửa công việc

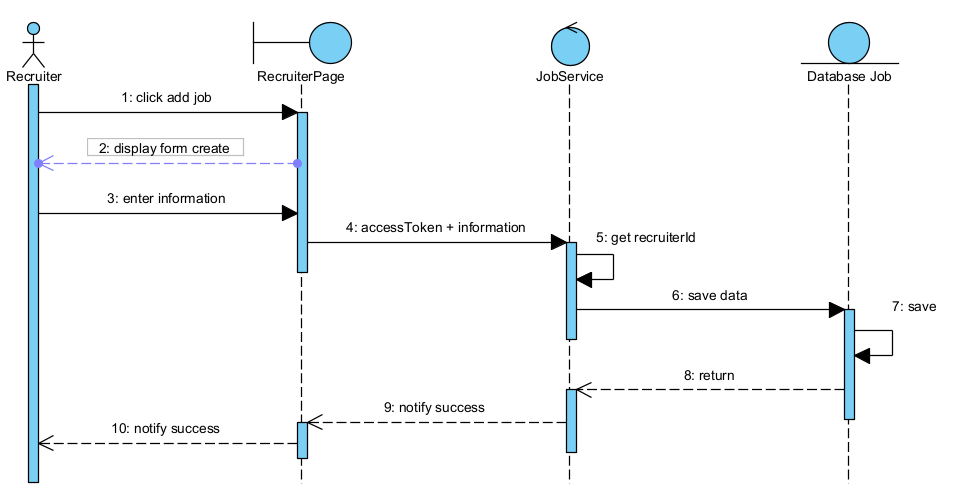
**Scenario thêm công việc chuẩn :**

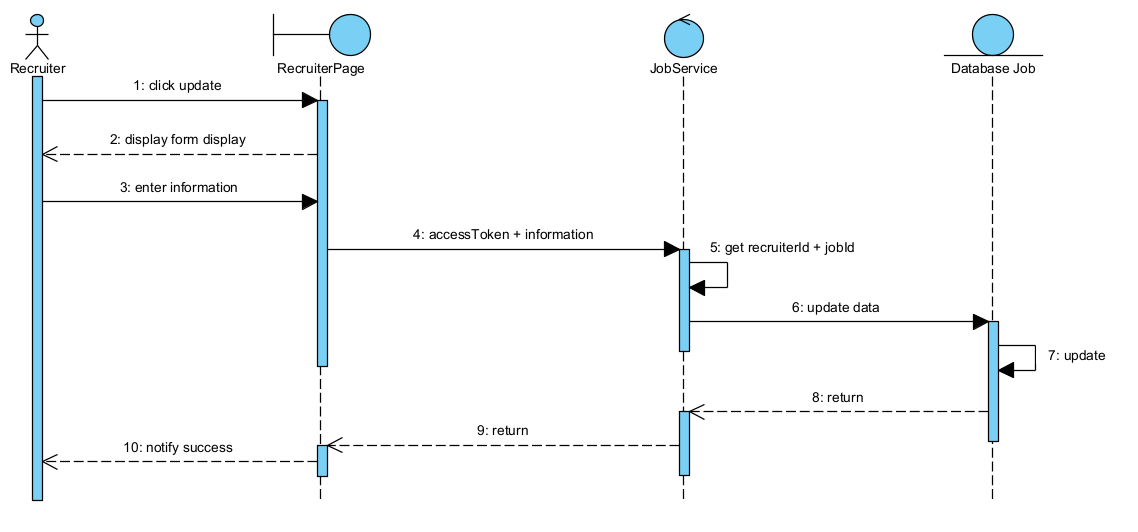
1. Recruiter nhấn vào nút "add job" trên RecruiterPage.
2. RecruiterPage hiển thị form tạo công việc.
3. Recruiter nhập thông tin công việc vào form và gửi đi.
4. RecruiterPage gửi accessToken kèm theo thông tin công việc đến JobService.
5. JobService sử dụng accessToken để lấy recruiterId.
6. JobService gửi yêu cầu lưu dữ liệu công việc đến Database Job.
7. Database Job thực hiện lưu thông tin công việc.
8. Database Job trả về kết quả lưu thành công cho JobService.
9. JobService gửi thông báo thành công về cho RecruiterPage.
10. RecruiterPage thông báo lại cho Recruiter là công việc đã được thêm thành công.

**Scenario sửa công việc chuẩn :**

1. Recruiter nhấn vào nút "update" trên giao diện RecruiterPage.
2. RecruiterPage hiển thị form cập nhật thông tin công việc.
3. Recruiter điền thông tin cập nhật vào form và gửi đi.
4. RecruiterPage gửi accessToken kèm thông tin mới đến JobService.
5. JobService sử dụng accessToken để lấy recruiterId và jobId tương ứng.
6. JobService gửi yêu cầu cập nhật dữ liệu công việc đến Database Job.
7. Database Job thực hiện cập nhật thông tin công việc.
8. Database Job trả về kết quả cập nhật cho JobService.
9. JobService gửi phản hồi lại cho RecruiterPage.
10. RecruiterPage hiển thị thông báo thành công cho Recruiter.

* Sơ đồ tuần tự

****

****

* **API tạo công việc :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Request** | | | |
| **Mô tả** | Lấy thông tin chi tiết công việc | | |
| **Enpoint** | /job/create | | |
| **Method** | POST | | |
| **Param** | Không | | |
| **Query** | Không | | |
| **Body** | **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| jobName | String | Tên công việc |
| minSalary | Double | Lương thấp nhất |
| maxSalary | Double | Lương cao nhất |
| Address | String | Địa chỉ |
| expirationDate | String | Ngày hết hạn |
| occupationName | String | Lĩnh vực |
| Experience | String | Kinh nghiệm |
| headCount | Int | Số lượng tuyển |
| jobType | String | Hình thức làm việc |
| JobLevel | String | Cấp bậc |
| educationLevel | String | Trình độ học vấn |
| Descriptions | String | Mô tả công việc |
| requiredJobList | String | Yêu cầu công việc |
| employeeBenefitList | String | Lợi ích công việc |
| **Response** | | | |
| **VD** |  | | |

* **API sửa công việc :**

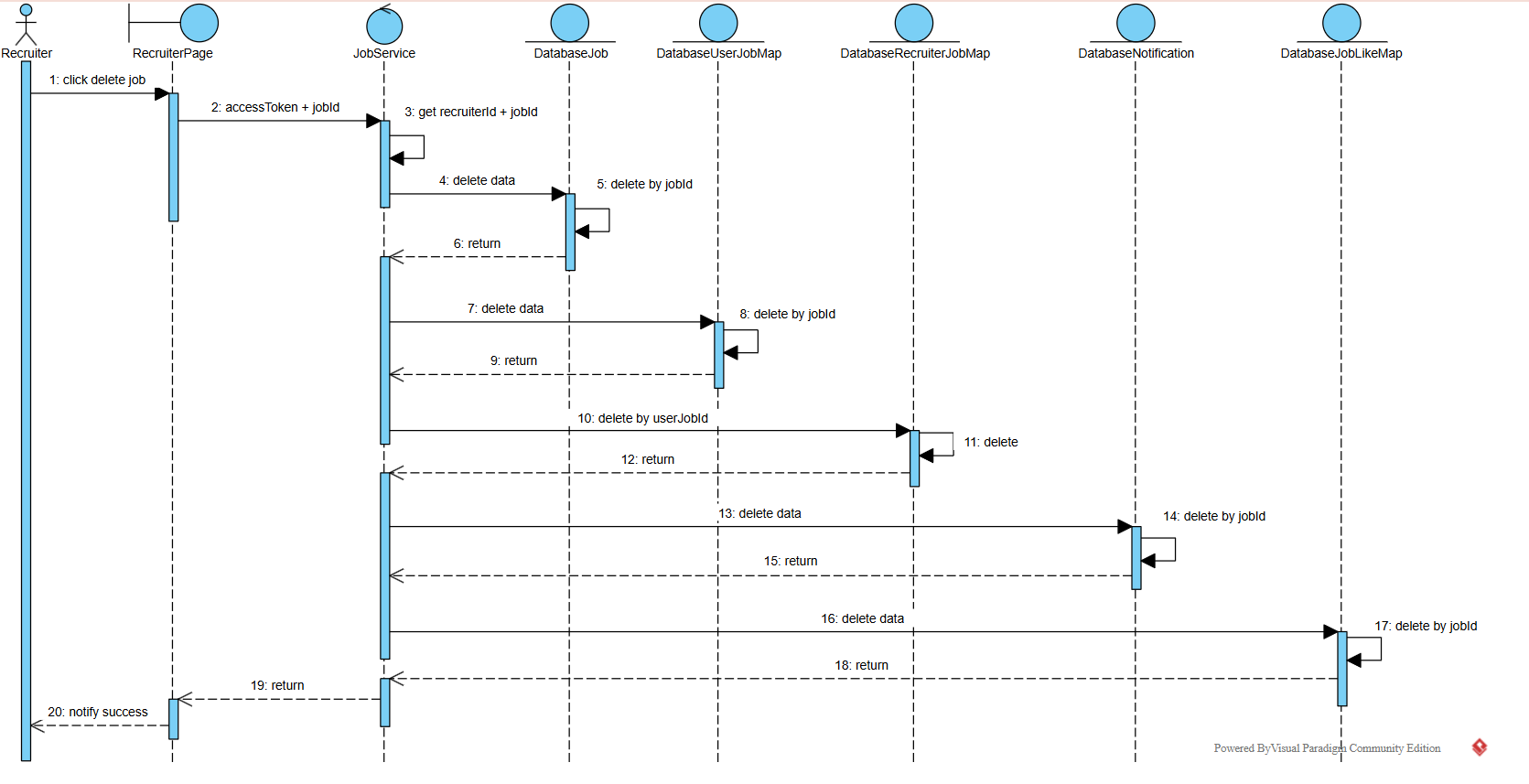
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Request** | | | | | |
| **Mô tả** | Sửa thông tin chi tiết công việc | | | | |
| **Enpoint** | /job/update | | | | |
| **Method** | POST | | | | |
| **Param** | **Tên trường** | **Kiểu dữ liệu** | | **Mô tả** | |
| jobId | Long | | Id công việc | |
| **Query** | Không | | | | |
| **Body** | **Tên trường** | | **Kiểu dữ liệu** | | **Mô tả** |
| jobName | | String | | Tên công việc |
| minSalary | | Double | | Lương thấp nhất |
| maxSalary | | Double | | Lương cao nhất |
| Address | | String | | Địa chỉ |
| expirationDate | | String | | Ngày hết hạn |
| occupationName | | String | | Lĩnh vực |
| Experience | | String | | Kinh nghiệm |
| headCount | | Int | | Số lượng tuyển |
| jobType | | String | | Hình thức làm việc |
| JobLevel | | String | | Cấp bậc |
| educationLevel | | String | | Trình độ học vấn |
| Descriptions | | String | | Mô tả công việc |
| requiredJobList | | String | | Yêu cầu công việc |
| employeeBenefitList | | String | | Lợi ích công việc |
| **Response** | | | | | |
| **VD** |  | | | | |

1. Xóa công việc

**Scenario chuẩn :**

1. Recruiter nhấn nút "delete job" trên giao diện.
2. RecruiterPage gửi accessToken và jobId đến JobService.
3. JobService xác thực và lấy recruiterId cùng jobId.
4. JobService bắt đầu tiến trình xoá dữ liệu.
5. JobService gửi yêu cầu xoá job theo jobId đến DatabaseJob.
6. DatabaseJob xoá và trả về kết quả cho JobService.
7. JobService gửi tiếp yêu cầu xoá dữ liệu liên quan trong DatabaseUserJobMap.
8. DatabaseUserJobMap thực hiện xoá theo jobId.
9. Trả về kết quả cho JobService.
10. JobService gửi yêu cầu xoá theo userJobId đến DatabaseRecruiterJobMap.
11. DatabaseRecruiterJobMap thực hiện xoá dữ liệu.
12. Trả về kết quả cho JobService.
13. JobService gửi yêu cầu xoá thông báo liên quan đến job trong DatabaseNotification.
14. DatabaseNotification xoá theo jobId.
15. Trả về kết quả cho JobService.
16. JobService tiếp tục gửi yêu cầu xoá dữ liệu lượt thích trong DatabaseJobLikeMap.
17. DatabaseJobLikeMap xoá theo jobId.
18. Trả về kết quả cho JobService.
19. JobService trả về kết quả tổng thể cho RecruiterPage.
20. RecruiterPage thông báo xoá thành công cho Recruiter.

* Dưới đây là sơ đồ luồng hoạt động :



* API :

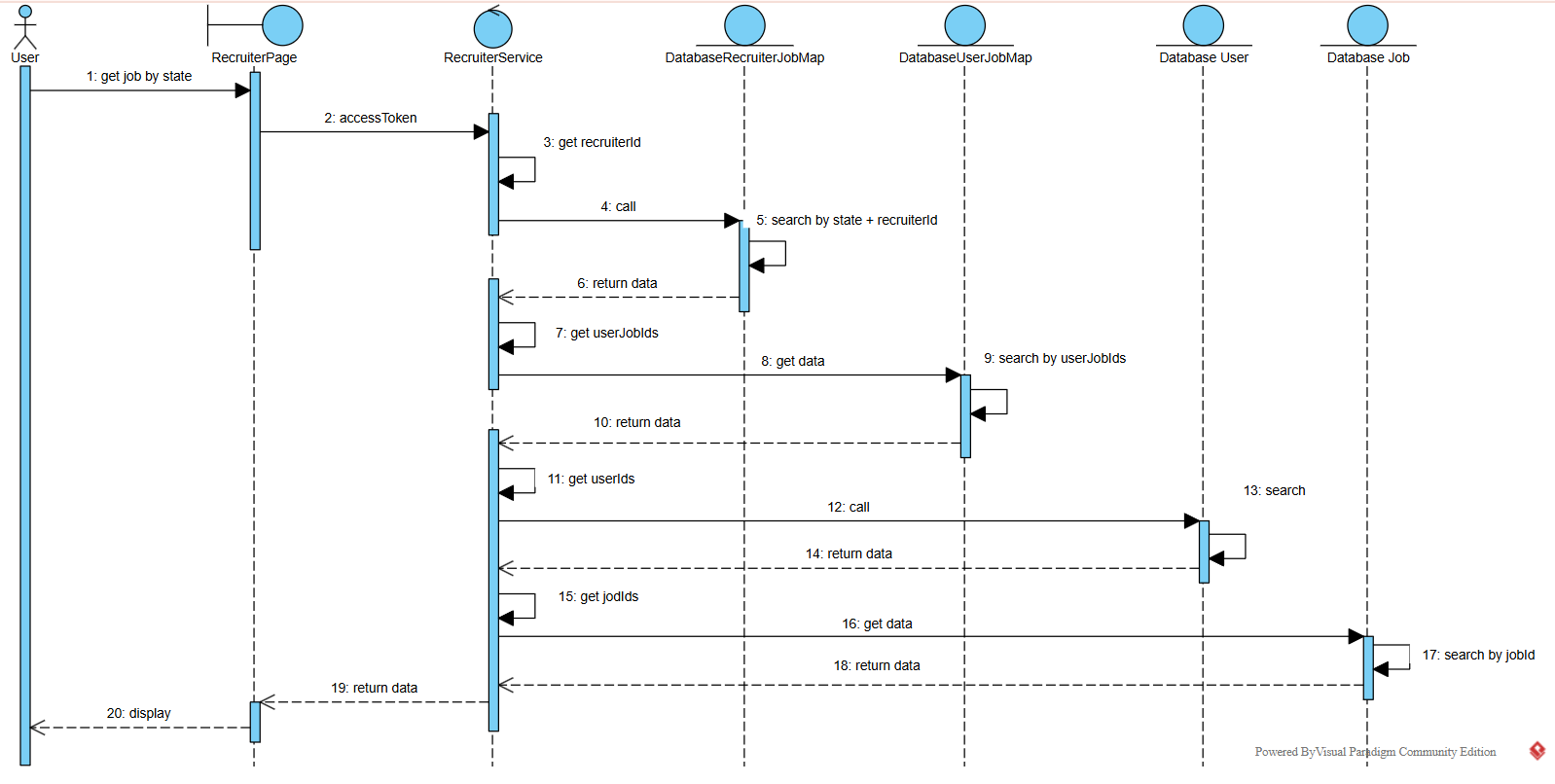
|  |  |  |  |
| --- | --- | --- | --- |
| **Request** | | | |
| **Mô tả** | Sửa thông tin chi tiết công việc | | |
| **Enpoint** | /job/delete | | |
| **Method** | DELETE | | |
| **Param** | **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| jobId | Long | Id công việc |
| **Query** | Không | | |
| **Body** | Không | | |
| **Response** | | | |
| **VD** |  | | |

1. Xem công việc ứng tuyển theo trạng thái

**Scenario chuẩn :**

1. User gửi yêu cầu "get job by state" đến RecruiterPage.
2. RecruiterPage gửi yêu cầu lấy accessToken đến RecruiterService.
3. RecruiterService lấy recruiterId từ accessToken.
4. RecruiterService gọi DatabaseRecruiterJobMap để tìm theo recruiterId và state.
5. DatabaseRecruiterJobMap thực hiện truy vấn "search by state + recruiterId".
6. DatabaseRecruiterJobMap trả dữ liệu về cho RecruiterService.
7. RecruiterService tiếp tục lấy userJobIds liên quan.
8. Gửi truy vấn đến DatabaseUserJobMap.
9. DatabaseUserJobMap thực hiện truy vấn "search by userJobIds".
10. DatabaseUserJobMap trả dữ liệu userJobIds cho RecruiterService.
11. RecruiterService gửi truy vấn lấy danh sách userIds.
12. Gọi đến Database User để lấy thông tin người dùng.
13. Database User thực hiện truy vấn "search".
14. Database User trả dữ liệu userIds cho RecruiterService.
15. RecruiterService gửi yêu cầu lấy danh sách jobIds từ userIds.
16. Gửi truy vấn đến DatabaseRecruiterJobMap.
17. DatabaseRecruiterJobMap truy vấn thông tin job theo jobId.
18. DatabaseRecruiterJobMap trả dữ liệu job.
19. RecruiterService trả toàn bộ dữ liệu đã tổng hợp cho RecruiterPage.
20. RecruiterPage hiển thị dữ liệu kết quả cho User.

* Sơ đồ luồng hoạt động :



* API :

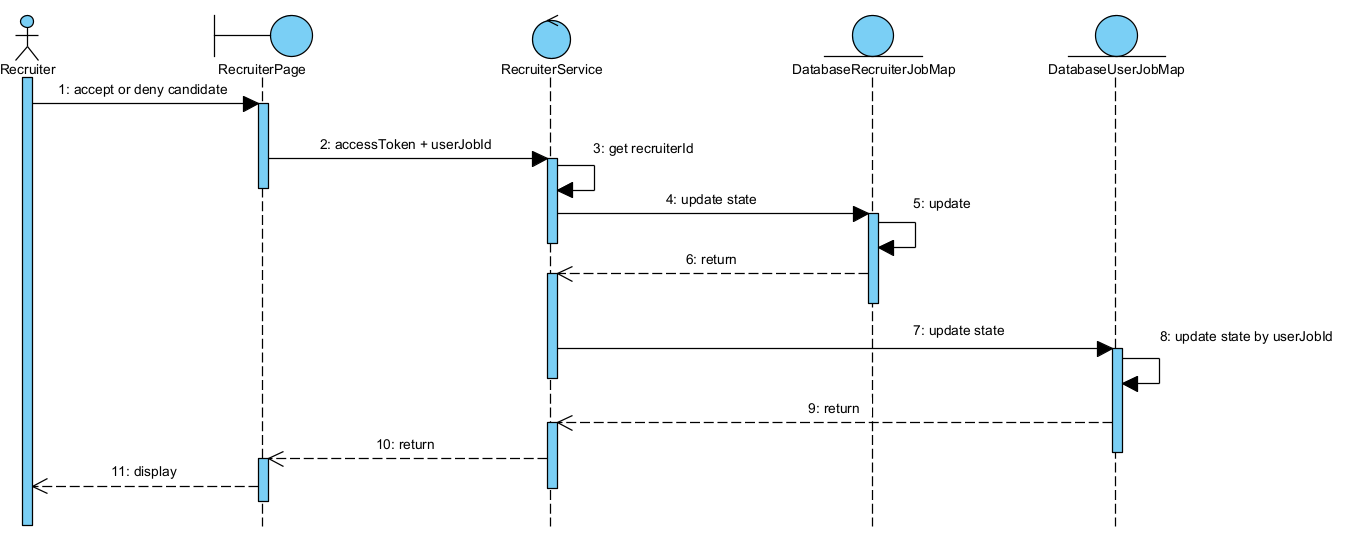
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Request** | | | | | |
| **Mô tả** | Lấy ra thông tin ứng tuyển theo state | | | | |
| **Enpoint** | /recruiter/state | | | | |
| **Method** | GET | | | | |
| **Param** | **Tên trường** | **Kiểu dữ liệu** | | **Mô tả** | |
| state | String | | Trạng thái đơn ứng tuyển | |
| page | Int | | Trang | |
| size | Int | | Kích thước trang | |
| **Query** | Không | | | | |
| **Body** | Không | | | | |
| **Response** | | | | | |
| **Kiểu** | Page<Object> | | | | |
| **Định dạng dữ liệu** | **Tên trường** | | **Kiểu dữ liệu** | | **Mô tả** |
| userOutput | | Object | | Thông tin ứng viên |
| cvUrl | | String | | CV |
| jobName | | String | | Tên công việc |
|  | State | | String | | Trạng thái hiện tại |
|  | applyDate | | Date | | Ngày nộp đơn |
| **VD** |  | | | | |

1. Chấp nhận hoặc từ chối ứng viên

**Scenario chuẩn :**

1. Recruiter chọn chấp nhận hoặc từ chối ứng viên trên RecruiterPage.
2. RecruiterPage gửi accessToken và userJobId đến RecruiterService.
3. RecruiterService sử dụng accessToken để lấy recruiterId tương ứng.
4. RecruiterService gửi yêu cầu cập nhật trạng thái đến DatabaseRecruiterJobMap.
5. DatabaseRecruiterJobMap thực hiện cập nhật trạng thái tương ứng (ví dụ: từ pending → accepted).
6. DatabaseRecruiterJobMap trả kết quả cập nhật về cho RecruiterService.
7. RecruiterService tiếp tục gửi yêu cầu cập nhật trạng thái tương ứng trong DatabaseUserJobMap.
8. DatabaseUserJobMap cập nhật trạng thái dựa theo userJobId.
9. DatabaseUserJobMap trả kết quả cập nhật về cho RecruiterService.
10. RecruiterService trả kết quả cuối cùng về cho RecruiterPage.
11. RecruiterPage hiển thị kết quả cập nhật trạng thái ứng viên cho Recruiter.

* Sơ đồ luồng hoạt động :



* API duyệt đơn :

|  |  |  |  |
| --- | --- | --- | --- |
| **Request** | | | |
| **Mô tả** | Duyệt ứng viên | | |
| **Enpoint** | /recruiter/accept | | |
| **Method** | POST | | |
| **Param** | **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| recruiterJobId | Long | Id bản ghi |
| **Query** | Không | | |
| **Body** | Không | | |
| **Response** | | | |
| **VD** |  | | |

* API từ chối :

|  |  |  |  |
| --- | --- | --- | --- |
| **Request** | | | |
| **Mô tả** | Từ chối ứng viên | | |
| **Enpoint** | /recruiter/reject | | |
| **Method** | POST | | |
| **Param** | **Tên trường** | **Kiểu dữ liệu** | **Mô tả** |
| recruiterJobId | Long | Id bản ghi |
| **Query** | Không | | |
| **Body** | Không | | |
| **Response** | | | |
| **VD** |  | | |

### Admin

1. Quản lý người dùng
2. Quản lý công việc

# KẾT QUẢ